日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 Date of Application:	２００５年　５月２５日
出願番号 Application Number:	特願２００５－１５２７８８
パリ条約による外国への出願に用いる優先権の主張の基礎となる出願の国コードと出願番号 The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is	ＪＰ２００５－１５２７８８
出願人 Applicant(s):	ソニー株式会社

２００５年１０月２６日

特許庁長官
Commissioner,
Japan Patent Office

中嶋誠

【書類名】　特許願
【整理番号】　0500017605
【提出日】　平成17年 5月25日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 15/16
【発明者】
　【住所又は居所】　東京都港区南青山二丁目６番２１号　株式会社ソニー・コンピュータエンタテインメント内
　【氏名】　鈴鹿　倫之
【特許出願人】
　【識別番号】　000002185
　【氏名又は名称】　ソニー株式会社
【代理人】
　【識別番号】　100093241
　【弁理士】
　【氏名又は名称】　宮田　正昭
　【電話番号】　03-5541-7577
【選任した代理人】
　【識別番号】　100101801
　【弁理士】
　【氏名又は名称】　山田　英治
　【電話番号】　03-5541-7577
【選任した代理人】
　【識別番号】　100086531
　【弁理士】
　【氏名又は名称】　澤田　俊夫
　【電話番号】　03-5541-7577
【先の出願に基づく優先権主張】
　【出願番号】　特願2004-283531
　【出願日】　平成16年 9月29日
【手数料の表示】
　【予納台帳番号】　048747
　【納付金額】　16,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9904833

【書類名】特許請求の範囲
【請求項１】
複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置であり、
複数ＯＳ間の通信制御を実行する制御ＯＳを有し、
前記制御ＯＳは、
通信実行ＯＳ各々に対応して設定される論理パーティション相互のメッセージ転送制御を実行する構成であり、物理アドレス空間のメッセージ領域を、メッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態から、メッセージ受信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態に切り替える処理を実行して通信実行ＯＳ間のメッセージ転送制御を行なう構成であることを特徴とする情報処理装置。
【請求項２】
前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、
自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介してアクセス可能な仮想ファイルを作成し、前記物理アドレス空間のメッセージ領域を該仮想ファイルを介してアクセスする構成であることを特徴とする請求項１に記載の情報処理装置。
【請求項３】
ソケットを適用した通信を実行する通信実行ＯＳは、
ソケットに対応付けられた仮想ファイルの識別子を取得し、取得した仮想ファイル識別子によって特定される仮想ファイルを適用したメッセージ書き込みまたはメッセージ読み取りを実行する構成であることを特徴とする請求項２に記載の情報処理装置。
【請求項４】
前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、
自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域をプロセスのアドレス空間へマッピングし、プロセスが直接メッセージ領域へのアクセスを実行する構成であることを特徴とする請求項１に記載の情報処理装置。
【請求項５】
前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、
自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域を前記通信実行ＯＳに対応する論理パーティションのアドレス空間へマッピングし、前記メッセージ領域へのアクセスを実行する構成であることを特徴とする請求項１に記載の情報処理装置。
【請求項６】
ソケットを適用した通信を実行する通信実行ＯＳは、
前記ソケットに対応するサービス識別子を設定し、サービス対応の通信許可設定を実行する構成であることを特徴とする請求項２乃至５いずれかに記載の情報処理装置。
【請求項７】
ソケットを適用した通信を実行する通信実行ＯＳは、
前記ソケットを介したメッセージの受信監視処理を実行する構成であることを特徴とする請求項２乃至５いずれかに記載の情報処理装置。
【請求項８】
前記通信実行ＯＳは、
システムコールとしてのセレクトシステムコールの適用により、前記ソケットを介したメッセージの受信監視処理を実行する構成であることを特徴とする請求項７に記載の情報処理装置。
【請求項９】
複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置における通信処理方

伝送あり、
物理アドレス空間のメッセージ領域を、メッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態とするステップと、
前記マッピング状態を解除し、物理アドレス空間のメッセージ領域を、メッセージ受信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態に変更するステップと、
を有することを特徴とする通信処理方法。
【請求項１０】
メッセージ転送を実行する少なくとも１つの通信実行ＯＳは、
自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介してアクセス可能な仮想ファイルを作成し、前記物理アドレス空間のメッセージ領域を該仮想ファイルを介してアクセスすることによるメッセージ送受信を行なうことを特徴とする請求項９に記載の通信処理方法。
【請求項１１】
ソケットを適用した通信を実行する通信実行ＯＳは、
ソケットに対応付けられた仮想ファイルの識別子を取得し、取得した仮想ファイル識別子によって特定される仮想ファイルを適用したメッセージ書き込みまたはメッセージ読み取りを実行することを特徴とする請求項１０に記載の通信処理方法。
【請求項１２】
前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、
自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域をプロセスのアドレス空間へマッピングし、プロセスが直接メッセージ領域へのアクセスを実行することを特徴とする請求項９に記載の通信処理方法。
【請求項１３】
前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、
自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域を前記通信実行ＯＳに対応する論理パーティションのアドレス空間へマッピングし、前記メッセージ領域へのアクセスを実行する構成であることを特徴とする請求項９に記載の通信処理方法。
【請求項１４】
ソケットを適用した通信を実行する通信実行ＯＳは、
前記ソケットに対応するサービス識別子を設定し、サービス対応の通信許可設定を実行することを特徴とする請求項９乃至１３いずれかに記載の通信処理方法。
【請求項１５】
ソケットを適用した通信を実行する通信実行ＯＳは、
前記ソケットを介したメッセージの受信監視処理を実行することを特徴とする請求項９乃至１３いずれかに記載の通信処理方法。
【請求項１６】
前記通信実行ＯＳは、
システムコールとしてのセレクトシステムコールの適用により、前記ソケットを介したメッセージの受信監視処理を実行することを特徴とする請求項１５に記載の通信処理方法。
【請求項１７】
複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置における通信処理制御を実行するコンピュータ・プログラムであり、
物理アドレス空間のメッセージ領域を、メッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態とするステップと、
前記マッピング状態を解除し、物理アドレス空間のメッセージ領域を、メッセージ受信

前ＯＳの論理パーティションアドレス空間のアクセス領域へのマッピング状態に変更するステップと、
を有することを特徴とするコンピュータ・プログラム。
【請求項１８】
複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置における通信処理制御を実行するコンピュータ・プログラムであり、
通信実行ＯＳの管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成するステップと、
前記ソケットを介してアクセス可能な仮想ファイルを作成するステップと、
前記物理アドレス空間のメッセージ領域を前記仮想ファイルを介してアクセスすることによってメッセージ送受信を行なうステップと、
を有することを特徴とするコンピュータ・プログラム。
【請求項１９】
複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置における通信処理制御を実行するコンピュータ・プログラムであり、
通信実行ＯＳの管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成するステップと、
前記ソケットを介して前記物理アドレス空間のメッセージ領域をプロセスのアドレス空間へマッピングするステップと、
プロセスが直接メッセージ領域へのアクセスを実行するステップと、
を有することを特徴とするコンピュータ・プログラム。

【書類名】明細書
【発明の名称】情報処理装置、通信処理方法、並びにコンピュータ・プログラム
【技術分野】
【０００１】
本発明は、情報処理装置、通信処理方法、並びにコンピュータ・プログラムに関する。さらに詳細には、複数のオペレーティングシステム（ＯＳ）によるプロセス実行を可能としたマルチＯＳ構成において、ＯＳ間のメッセージ転送制御を実行する情報処理装置、通信処理方法、並びにコンピュータ・プログラムに関する。
【背景技術】
【０００２】
１つのシステム内に複数のオペレーティングシステム（ＯＳ）を搭載したマルチＯＳシステムにおいては、各ＯＳはそれぞれ異なるプロセスが実行可能であり、システムで共通のハードウェア、すなわちＣＰＵやメモリ等を時系列に順次切り替えて利用した処理が行なわれる。
【０００３】
複数ＯＳの各々の実行プロセス（タスク）のスケジューリングは、例えばパーティション管理ソフトウェアによって実行される。１つのシステムにＯＳ（α）とＯＳ（β）の２つのオペレーティングシステムが並存する場合、ＯＳ（α）の処理をパーティションＡとし、ＯＳ（β）の処理をパーティションＢとすると、パーティション管理ソフトウェアは、パーティションＡとパーティションＢの実行スケジュールを決定し、決定したスケジュールに基づいて、ハードウェア資源を割り当てて各ＯＳにおける処理を実行する。
【０００４】
マルチＯＳ型のシステムにおけるタスク管理を開示した従来技術として、例えば、特許文献１がある。特許文献１には、複数のＯＳの各々において実行されるタスク管理において、緊急性の高い処理を優先的に処理させるためのタスクスケジューリング手法を開示している。
【０００５】
複数ＯＳが共存するシステムでは、ＯＳ間のメッセージ転送を行なう場合、ＯＳ間を接続するメッセージチャネルを生成し、通信を実行するＯＳ各々がメッセージチャネル接続用の通信用インタフェースを設定することが必要となる。メッセージチャネルを介した通信を実行する通信実行ＯＳが様々な異なるＯＳである場合、通信インタフェース仕様の制限や、異なるＯＳ間通信に起因するオーバヘッドの発生など、様々な問題がある。
【特許文献１】特開２００３－３４５６１２号公報
【発明の開示】
【発明が解決しようとする課題】
【０００６】
本発明は、上述の問題点に鑑みてなされたものであり、複数ＯＳが共存するマルチＯＳ環境におけるＯＳ間通信を余分なオーバヘッドを発生させることなくスムーズに実行することを可能とした情報処理装置、通信処理方法、並びにコンピュータ・プログラムを提供することを目的とする。
【課題を解決するための手段】
【０００７】
本発明の第１の側面は、
複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置であり、
複数ＯＳ間の通信制御を実行する制御ＯＳを有し、
前記制御ＯＳは、
通信実行ＯＳ各々に対応して設定される論理パーティション相互のメッセージ転送制御を実行する構成であり、物理アドレス空間のメッセージ領域を、メッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態から、メッセージ受信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態に切

り替える処理を実行して通信実行ＯＳ間のメッセージ転送制御を行なう構成であることを特徴とする情報処理装置にある。

【０００８】

さらに、本発明の情報処理装置の一実施態様において、前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介してアクセス可能な仮想ファイルを作成し、前記物理アドレス空間のメッセージ領域を該仮想ファイルを介してアクセスする構成であることを特徴とする。

【０００９】

さらに、本発明の情報処理装置の一実施態様において、ソケットを適用した通信を実行する通信実行ＯＳは、ソケットに対応付けられた仮想ファイルの識別子を取得し、取得した仮想ファイル識別子によって特定される仮想ファイルを適用したメッセージ書き込みまたはメッセージ読み取りを実行する構成であることを特徴とする。

【００１０】

さらに、本発明の情報処理装置の一実施態様において、前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域をプロセスのアドレス空間へマッピングし、プロセスが直接メッセージ領域へのアクセスを実行する構成であることを特徴とする。

【００１１】

さらに、本発明の情報処理装置の一実施態様において、前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域を前記通信実行ＯＳに対応する論理パーティションのアドレス空間へマッピングし、前記メッセージ領域へのアクセスを実行する構成であることを特徴とする。

【００１２】

さらに、本発明の情報処理装置の一実施態様において、ソケットを適用した通信を実行する通信実行ＯＳは、前記ソケットに対応するサービス識別子を設定し、サービス対応の通信許可設定を実行する構成であることを特徴とする。

【００１３】

さらに、本発明の情報処理装置の一実施態様において、ソケットを適用した通信を実行する通信実行ＯＳは、前記ソケットを介したメッセージの受信監視処理を実行する構成であることを特徴とする。

【００１４】

さらに、本発明の情報処理装置の一実施態様において、前記通信実行ＯＳは、システムコールとしてのセレクトシステムコールの適用により、前記ソケットを介したメッセージの受信監視処理を実行する構成であることを特徴とする。

【００１５】

さらに、本発明の第２の側面は、

複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置における通信処理方法であり、

物理アドレス空間のメッセージ領域を、メッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態とするステップと、

前記マッピング状態を解除し、物理アドレス空間のメッセージ領域を、メッセージ受信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態に変更するステップと、

を有することを特徴とする通信処理方法にある。

【００１６】

さらに、本発明の通信処理方法の一実施態様において、メッセージ転送を実行する少な

くとも１つの通信実行ＯＳは、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介してアクセス可能な仮想ファイルを作成し、前記物理アドレス空間のメッセージ領域を該仮想ファイルを介してアクセスすることによるメッセージ送受信を行なうことを特徴とする。

【００１７】

さらに、本発明の通信処理方法の一実施態様において、ソケットを適用した通信を実行する通信実行ＯＳは、ソケットに対応付けられた仮想ファイルの識別子を取得し、取得した仮想ファイル識別子によって特定される仮想ファイルを適用したメッセージ書き込みまたはメッセージ読み取りを実行することを特徴とする。

【００１８】

さらに、本発明の通信処理方法の一実施態様において、前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域をプロセスのアドレス空間へマッピングし、プロセスが直接メッセージ領域へのアクセスを実行することを特徴とする。

【００１９】

さらに、本発明の通信処理方法の一実施態様において、前記通信実行ＯＳ中の少なくとも１つの通信実行ＯＳは、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域を前記通信実行ＯＳに対応する論理パーティションのアドレス空間へマッピングし、前記メッセージ領域へのアクセスを実行する構成であることを特徴とする。

【００２０】

さらに、本発明の通信処理方法の一実施態様において、ソケットを適用した通信を実行する通信実行ＯＳは、前記ソケットに対応するサービス識別子を設定し、サービス対応の通信許可設定を実行することを特徴とする。

【００２１】

さらに、本発明の通信処理方法の一実施態様において、ソケットを適用した通信を実行する通信実行ＯＳは、前記ソケットを介したメッセージの受信監視処理を実行することを特徴とする。

【００２２】

さらに、本発明の通信処理方法の一実施態様において、前記通信実行ＯＳは、システムコールとしてのセレクトシステムコールの適用により、前記ソケットを介したメッセージの受信監視処理を実行することを特徴とする。

【００２３】

さらに、本発明の第３の側面は、

複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置における通信処理制御を実行するコンピュータ・プログラムであり、

物理アドレス空間のメッセージ領域を、メッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態とするステップと、

前記マッピング状態を解除し、物理アドレス空間のメッセージ領域を、メッセージ受信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態に変更するステップと、

を有することを特徴とするコンピュータ・プログラムにある。

【００２４】

さらに、本発明の第４の側面は、

複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置における通信処理制御を実行するコンピュータ・プログラムであり、

通信実行ＯＳの管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成するステップと、

前記ソケットを介してアクセス可能な仮想ファイルを作成するステップと、
前記物理アドレス空間のメッセージ領域を前記仮想ファイルを介してアクセスすることによってメッセージ送受信を行なうステップと、
を有することを特徴とするコンピュータ・プログラムにある。
【００２５】
さらに、本発明の第５の側面は、
複数のオペレーティングシステム（ＯＳ）を格納した情報処理装置における通信処理制御を実行するコンピュータ・プログラムであり、
通信実行ＯＳの管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成するステップと、
前記ソケットを介して前記物理アドレス空間のメッセージ領域をプロセスのアドレス空間へマッピングするステップと、
プロセスが直接メッセージ領域へのアクセスを実行するステップと、
を有することを特徴とするコンピュータ・プログラムにある。
【００２６】
なお、本発明のコンピュータ・プログラムは、例えば、様々なプログラム・コードを実行可能な汎用コンピュータ・システムに対して、コンピュータ可読な形式で提供する記憶媒体、通信媒体、例えば、ＣＤやＦＤ、ＭＯなどの記憶媒体、あるいは、ネットワークなどの通信媒体によって提供可能なコンピュータ・プログラムである。このようなプログラムをコンピュータ可読な形式で提供することにより、コンピュータ・システム上でプログラムに応じた処理が実現される。
【００２７】
本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づく、より詳細な説明によって明らかになるであろう。なお、本明細書においてシステムとは、複数の装置の論理的集合構成であり、各構成の装置が同一筐体内にあるものには限らない。
【発明の効果】
【００２８】
本発明の構成によれば、複数のオペレーティングシステム（ＯＳ）が共存するマルチＯＳ環境におけるＯＳ間通信において、ＯＳ間の通信制御を実行する制御ＯＳを設定し、制御ＯＳが、通信実行ＯＳ各々に対応して設定される論理パーティション相互のメッセージ転送制御を実行する構成とし、物理アドレス空間のメッセージ領域を、メッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態から、メッセージ受信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態に切り替える処理を実行してＯＳ間のメッセージ転送制御を行なう構成としたので、異なるＯＳ間の通信におけるスムーズな通信が可能となる。
【００２９】
さらに、本発明の構成によれば、通信実行ＯＳが、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、ソケットを介してアクセス可能な仮想ファイルを作成し、物理アドレス空間のメッセージ領域を該仮想ファイルを介してアクセスすることで、メッセージ送受信を実行する構成としたので、異なるＯＳとの通信において汎用的なソケットを適用したメッセージパッシングが実現される。
【００３０】
さらに、本発明の構成によれば、通信実行ＯＳが、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して前記物理アドレス空間のメッセージ領域をプロセスのアドレス空間へマッピングし、プロセスが直接メッセージ領域へのアクセスを実行する構成としたので、異なるＯＳとの通信において汎用的なソケットを適用したメッセージパッシングが実現されると同時に、少ないオーバヘッドでプロセスがメッセージ領域にアクセスすることが実現される

【００３１】
さらに、本発明の構成によれば、ソケットに対応するサービス識別子を設定することにより、サービス対応の通信許可設定が可能となり、また、例えばシステムコールとしてのセレクトシステムコールの適用により、ソケットを介したメッセージの受信監視処理が可能となる。

【発明を実施するための最良の形態】

【００３２】
以下、図面を参照しながら、本発明の情報処理装置、通信処理方法、並びにコンピュータ・プログラムの詳細について説明する。

【００３３】
まず、図１を参照して、本発明の情報処理装置のハードウェア構成例について説明する。プロセッサモジュール１０１は、複数のプロセッサ（Processing Unit）から構成されたモジュールであり、ＲＯＭ(Read Only Memory)１０４、ＨＤＤ１２３等に記憶されているプログラムに従って、オペレーティングシステム（ＯＳ：Operating System)、ＯＳ対応のアプリケーション・プログラムなど、各種プログラムに従ったデータ処理を実行する。プロセッサモジュール１０１の詳細については、さらに、後段で、図２を参照して説明する。

【００３４】
グラフィックエンジン１０２は、プロセッサモジュール１０１から入力される指示に従って、出力部１２２を構成する表示デバイスに画面出力するためのデータ生成、例えば３Ｄグラフィック描画処理を実行する。メインメモリ（ＤＲＡＭ）１０３には、プロセッサモジュール１０１において実行するプログラムやその実行において適宜変化するパラメータ等を格納する。これらはＣＰＵバスなどから構成されるホストバス１１１により相互に接続されている。

【００３５】
ホストバス１１１は、ブリッジ１０５を介して、ＰＣＩ(Peripheral Component Interconnect/Interface)バスなどの外部バス１１２に接続されている。ブリッジ１０５は、ホストバス１１１、外部バス１１２間、およびコントローラ１０６、メモリカード１０７、その他のデバイスとのデータ入出力制御を実行する。

【００３６】
入力部１２１は、キーボード、ポインティングデバイスなどのユーザにより操作される入力デバイスからの入力情報を入力する。出力部１２２は、液晶表示装置やＣＲＴ（Cathode Ray Tube）などの画像出力部とスピーカ等からなる音声出力部から構成される。

【００３７】
ＨＤＤ（Hard Disk Drive）１２３は、ハードディスクを内蔵し、ハードディスクを駆動し、プロセッサモジュール１０１によって実行するプログラムや情報を記録または再生させる。

【００３８】
ドライブ１２４は、装着されている磁気ディスク、光ディスク、光磁気ディスク、または半導体メモリ等のリムーバブル記録媒体１２７に記録されているデータまたはプログラムを読み出して、そのデータまたはプログラムを、インタフェース１１３、外部バス１１２、ブリッジ１０５、およびホストバス１１１を介して接続されているメインメモリ（ＤＲＡＭ）１０３に供給する。

【００３９】
接続ポート１２５は、外部接続機器１２８を接続するポートであり、ＵＳＢ，ＩＥＥＥ１３９４等の接続部を持つ。接続ポート１２５は、インタフェース１１３、外部バス１１２、ブリッジ１０５、およびホストバス１１１を介してプロセッサモジュール１０１等に接続されている。通信部１２６は、ネットワークに接続され、プロセッサモジュール１０１や、ＨＤＤ１２３等から提供されるデータの送信、外部からのデータ受信を実行する。

【００４０】
次に、プロセッサモジュールの構成例について、図２を参照して説明する。図２に示すように、プロセッサモジュール２００は、複数のメインプロセッサからなるメインプロセッサグループ２０１、複数のサブプロセッサからなる複数のサブプロセッサグループ２０２～２０ｎによって構成されている。それぞれにメモリコントローラ、２次キャッシュが設けられている。各プロセッサグループ２０１～２０ｎの各々は例えば８つのプロセッサユニットを有し、クロスバーアーキテクチャ、あるいはパケット交換式ネットワークなどによって接続されている。メインプロセッサグループ２０１のメインプロセッサの指示のもとに、複数のサブプロセッサグループ２０２～２０ｎの１以上のサブプロセッサが選択され、所定のプログラムが実行される。
【００４１】
各プロセッサグループに設置されたメモリフローコントローラは、図１に示すメインメモリ１０３とのデータ入出力制御を実行し、２次キャッシュは、各プロセッサグループにおける処理用データの記憶領域として利用される。
【００４２】
次に、図３を参照して、本発明の情報処理装置におけるオペレーティングシステム（ＯＳ）構成を説明する。本発明の情報処理装置は複数のオペレーティングシステム（ＯＳ）が共存するマルチＯＳ構成を持つ。図３に示すように、論理的な階層構成を持つ複数のオペレーティングシステム（ＯＳ）を持つ。
【００４３】
図３に示すように、下位レイヤに制御ＯＳ３０１を有し、上位レイヤに複数のゲストＯＳ３０２，３０３、およびシステム制御ＯＳ３０４が設定される。制御ＯＳ３０１は、システム制御ＯＳ３０４とともに図１、図２を参照して説明したプロセッサモジュールにおいて実行する１以上のプロセスによって適用される論理パーティションを実現し、システム内のハードウェア資源（計算機資源としてのメインプロセッサ、サブプロセッサ、メモリ、デバイス等）を各論理パーティションに割り振る処理を実行する。
【００４４】
ゲストＯＳ３０２，３０３は、例えばゲームＯＳやＷｉｎｄｏｗｓ（登録商標）、Ｌｉｎｕｘ（登録商標）などの各種のＯＳであり、制御ＯＳ３０１の制御の下に動作するＯＳである。なお、図３には、ゲストＯＳ３０２，３０３の２つのゲストＯＳのみを示してあるが、ゲストＯＳは任意の数に設定することが可能である。
【００４５】
ゲストＯＳ３０２，３０３は、制御ＯＳ３０１およびシステム制御ＯＳ３０４によって設定された論理パーティション内で動作し、その論理パーティションに割り当てられたメインプロセッサ、サブプロセッサ、メモリ、デバイス等のハードウェア資源を適用して各種のデータ処理を実行する。
【００４６】
例えば、ゲストＯＳ（ａ）３０２は、制御ＯＳ３０１およびシステム制御ＯＳ３０４によって設定された論理パーティション２に割り当てられたメインプロセッサ、サブプロセッサ、メモリ、デバイス等のハードウェア資源を適用して、ゲストＯＳ（ａ）３０２対応のアプリケーション・プログラム３０５を実行する。また、ゲストＯＳ（ｂ）３０３は、論理パーティションｎに割り当てられたメインプロセッサ、サブプロセッサ、メモリ、デバイス等のハードウェア資源を適用して、ゲストＯＳ（ｂ）３０３対応のアプリケーション・プログラム３０６を実行する。制御ＯＳ３０１は、ゲストＯＳの実行に必要なインタフェースとしてゲストＯＳプログラミングインタフェースを提供する。
【００４７】
システム制御ＯＳ３０４は、論理パーティション管理情報を含むシステム制御プログラム３０７を生成し、システム制御プログラム３０７に基づくシステムの動作制御を制御ＯＳ３０１とともに実行する。システム制御プログラム３０７は、システム制御プログラム・プログラミングインタフェースを用いてシステムのポリシを制御するプログラムである

。システム制御プログラム・プログラミングインタフェースは、制御ＯＳ３０１からシステム制御ＯＳ３０４に提供される。例えばリソース配分の上限値を設定するなど、プログラムによる柔軟なカスタマイズのための手段を提供するのがシステム制御プログラム３０７の役割である。

【００４８】

システム制御プログラム３０７はシステム制御プログラム・プログラミングインタフェースを用いてシステムの振る舞いを制御することができる。例えば、新しく論理パーティションを作成し、その論理パーティションで新しいゲストＯＳを起動することができる。複数のゲストＯＳが動作するシステムでは、それらのゲストＯＳはシステム制御プログラムにあらかじめプログラムされた順序で起動されることになる。また、ゲストＯＳから提出された資源割り当て要求を制御ＯＳ３０１が受理する前に検査し、システムのポリシに従って修正したり、要求そのものを拒否したりすることができる。これにより、特定のゲストＯＳだけが資源を独占することがないようにすることができる。このように、システムのポリシをプログラムとして実現したものがシステム制御プログラムである。

【００４９】

制御ＯＳ３０１はシステム制御ＯＳ３０４のために特別な論理パーティション（図では論理パーティション１）を割り当てる。制御ＯＳ３０１は、ハイパバイザモードで動作する。ゲストＯＳはスーパバイザモードで動作する。システム制御ＯＳ、アプリケーション・プログラムはプロブレムモード（ユーザモード）で動作する。

【００５０】

論理パーティションはシステム内の資源分配を受ける主体である。たとえばメインメモリ１０３（図１参照）はいくつかの領域へ分割され、それぞれの領域の使用権が論理パーティションに対して与えられる。論理パーティションに分配される資源の種別には以下のものがある。

ａ）物理プロセッサユニット使用時間
ｂ）仮想アドレス空間
ｃ）論理パーティション内で動作するプログラムがアクセスできるメモリ
ｄ）制御ＯＳが論理パーティションの管理のために用いるメモリ
ｅ）イベントポート
ｆ）デバイスの使用権
ｇ）キャッシュパーティション
ｈ）バス使用権

【００５１】

前述したように、ゲストＯＳは論理パーティションの中で動作する。ゲストＯＳは論理パーティションに割り当てられた資源を独占して各種のデータ処理を実行する。多くの場合、システム上で動作する個々のゲストＯＳ毎に１つの論理パーティションが作成される。各論理パーティションにはユニークな識別子が与えられる。システム制御ＯＳ３０４は、論理パーティション管理情報として生成するシステム制御プログラムに識別子を対応づけて管理する。

【００５２】

論理パーティションは、制御ＯＳ３０１およびシステム制御ＯＳ３０４によって生成される。生成直後の論理パーティションは何も資源を持たず、使用資源の制限も設定されていない。論理パーティションには活動状態と終了状態という２つの状態がある。生成直後の論理パーティションは活動状態にある。論理パーティション内で動作するゲストＯＳの要求に基づいて論理パーティションは終了状態に遷移し、論理パーティションに割り当てられている全ての論理プロセッサが停止する。

【００５３】

なお、論理プロセッサは、論理パーティションに割り当てられる論理的なプロセッサであり、いずれかの物理プロセッサ、すなわち、図２に示すプロセッサグループ内のプロセッサに対応する。ただし、論理プロセッサと物理プロセッサは必ずしも１対１の関係には

なく、１つの論理プロセッサに複数の物理プロセッサが対応付けられる場合もあり、複数の論理プロセッサに１つの物理プロセッサが対応付けられる場合もある。論理プロセッサと物理プロセッサの対応付けは、制御ＯＳ３０１が決定する。

【００５４】

制御ＯＳ３０１は、各論理パーティションが使用する資源の量を制限する機能を備えている。ゲストＯＳ３０２，３０３がシステム制御ＯＳ３０４との通信を行うことなく割り当て／解放が行える資源については使用量の制限が可能となっている。

【００５５】

各論理パーティションは制御シグナルポートを備えている。このポートには論理パーティション間のデータ交換／共有に必要な様々な制御シグナルが到着する。制御シグナルの例を以下に挙げる。

ａ）論理パーティション間イベントポートの接続依頼

ｂ）論理パーティション間メッセージチャネルの接続依頼

ｃ）共有メモリ領域への接続依頼

【００５６】

各論理パーティションに到着した制御シグナルは制御シグナルポートでキューイングされる。キューの深さは、メモリ資源が許す範囲であれば、制限は無い。キューイングに必要なメモリ資源は制御シグナルを送った論理パーティションから確保される。このポートから制御シグナルを取り出すためには、ゲストＯＳプログラミングインタフェースを呼び出す。空の制御シグナルポートに制御シグナルが到着したときに、任意のイベントポートにイベントを送信することが可能である。イベントポートの指定はゲストＯＳプログラミングインタフェースを呼び出すことによって行える。

【００５７】

次に、図４以下の各図を参照して、ＯＳ間通信について説明する。前述したように、各ゲストＯＳ、およびシステム制御ＯＳの各々に対しては、論理パーティションが対応付けられる。各論理パーティションモジュール間のメッセージ転送は、制御ＯＳの生成するメッセージチャネルを介して実行される。以下、

（１）ゲストＯＳ間の通信処理

（２）ゲストＯＳとシステム制御ＯＳ間の通信処理

の２つの通信処理について順次説明する。

【００５８】

［（１）ゲストＯＳ間の通信処理］

まず、ゲストＯＳ間の通信処理について説明する。図４は、制御ＯＳ４１０上の２つのゲストＯＳ間の通信処理を説明する概念図である。ゲストＯＳ（Ａ）４２０からゲストＯＳ（Ｂ）４３０にメッセージを転送する処理例を示している。

【００５９】

制御ＯＳ４１０は、ゲストＯＳ間の通信、あるいはゲストＯＳとシステム制御ＯＳ間の通信用のメッセージチャネル４１２を設定する。メッセージチャネルは２つの論理パーティション間のデータ転送機構として設定されるものであり、コネクションオリエンテッドな双方向の通信路として設定される。

【００６０】

図４を参照して、制御ＯＳ４１０の設定するメッセージチャネル４１２を介したゲストＯＳ（Ａ）４２０とゲストＯＳ（Ｂ）４３０間の通信処理について説明する。ゲストＯＳ（Ａ）４２０をメッセージチャネルのクライアント（接続元）、ゲストＯＳ（Ｂ）４３０をメッセージチャネルのサーバ（接続先）とする。

ゲストＯＳ（Ａ）４２０はメッセージチャネルの接続要求をゲストＯＳ（Ｂ）４３０に対して行う。

ゲストＯＳ（Ｂ）がメッセージチャネルの接続を許可すると、図４に示すように、制御ＯＳ４１０の制御の下にメッセージチャネル４１２がゲストＯＳ（Ａ）４２０とゲストＯＳ（Ｂ）４３０の間に接続される。メッセージチャネル４１２は、通信実行ＯＳとしての

各ゲストＯＳに設定される仮想的なアドレス空間としてのメッセージポート４２２，４３２を接続するチャネルとして設定される。

【００６１】

なお、ゲストＯＳ（Ｂ）４３０がメッセージチャネルの接続を拒否すると、メッセージチャネルの接続は行われない。各ゲストＯＳに対応するメッセージポート４２２，４３２にはイベント受信ポート４２１，４３１が関連づけて設定される。

【００６２】

制御ＯＳ４１０は、ＯＳ（論理パーティション）間通信、およびＯＳ（論理パーティション）内通信のためにイベント機構を用意している。イベントはイベント送信ポートからイベント受信ポートへの１対１、かつ単方向のコネクションを通して送られる。イベント受信ポートはイベントをキューイング及びカウントしない。例えばイベント受信ポートに読み出される前のイベントがある状態で新たなイベントが到着した場合、イベント受信ポートの状態はペンディング状態から変化しない。コネクションが確立されたイベント送信ポートは何度でもイベントを送信することができる。イベント受信ポートは論理パーティション内のゲストＯＳがイベントを受け取るためのポートである。イベント受信ポートに到着するイベントは割り込みの一つとして扱われる。

【００６３】

ＯＳ（論理パーティション）間にメッセージチャネルが接続されたり、空のメッセージポートにメッセージが到着すると、関連づけられたイベント受信ポートにイベントが配送される。

【００６４】

ゲストＯＳ（Ｂ）がメッセージチャネルの接続を許可すると、図４に示すように、制御ＯＳ４１０の制御の下にメッセージチャネル４１２がゲストＯＳ（Ａ）４２０とゲストＯＳ（Ｂ）４３０の間に接続され、その後、ゲストＯＳ（Ａ）４２０は送信するメッセージ領域を用意する。なお、メッセージ領域実体としての物理アドレス空間メッセージ領域４１１はシステムのメモリ上にある。

【００６５】

図４に示すゲストＯＳ（Ａ）４２０のメッセージ領域４２３と、ゲストＯＳ（Ｂ）４３０のメッセージ領域４３３は、それぞれのＯＳに対応する論理パーティションのアドレス空間における論理アドレス空間メッセージ領域である。ある瞬間に、物理アドレス空間メッセージ領域４１１はただ一つの論理パーティションに属している。物理アドレス空間メッセージ領域４１１が属していない論理パーティションからのメッセージ領域へのアクセスはできない。

【００６６】

すなわち、物理アドレス空間メッセージ領域４１１は、ある瞬間において、ゲストＯＳ（Ａ）４２０の論理アドレス空間メッセージ領域４２３にマッピングされた状態にあるか、あるいは、ゲストＯＳ（Ｂ）４３０の論理アドレス空間メッセージ領域４３３にマッピングされた状態にあるかいずれか一方の状態にある。このマッピング制御は制御ＯＳ４１０によって実行される。このように、制御ＯＳ４１０は、メッセージチャネル４１２を介したメッセージ転送を論理パーティションアドレス空間から物理アドレス空間への仮想的なアドレス変換機構を用いて実現する。

【００６７】

図５を参照して、仮想的なアドレス変換機構を用いたメッセージ転送処理について説明する。

【００６８】

図５において、（１）状態１、（２）状態２は、ゲストＯＳ（Ａ）からゲストＯＳ（Ｂ）に対するメッセージ転送の前後の状態を示している。（１）状態１において、物理アドレス空間４５０に確保されたメッセージ領域４５１は、メッセージ転送元（クライアント）側の論理パーティションの論理パーティションアドレス空間、すなわち、ゲストＯＳ（Ａ）の論理パーティションアドレス空間４６０のメッセージ領域４６１にマップされてい

る。この状態において、メッセージ領域４５１が属していない論理パーティション、すなわち、ゲストＯＳ（Ｂ）の論理パーティションアドレス空間４７０のメッセージ領域４７１から物理アドレス空間４５０のメッセージ領域４５１へのアクセスはできない。

【００６９】

メッセージ領域４５１が転送されると、物理アドレス空間４５０に確保されたメッセージ領域４５１は、クライアント側論理パーティション、すなわち、ゲストＯＳ（Ａ）の論理パーティションアドレス空間４６０からアンマップ（マップ解除）され、サーバ側論理パーティションの論理パーティションアドレス空間、すなわち、ゲストＯＳ（Ｂ）の論理パーティションアドレス空間４７０のメッセージ領域４７１にマップされる。すなわち、図５の（２）状態２に設定される。この状態において、メッセージ領域４５１が属していない論理パーティション、すなわち、ゲストＯＳ（Ａ）の論理パーティションアドレス空間４６０から物理アドレス空間４５０のメッセージ領域４５１へのアクセスはできない。

【００７０】

このように、制御ＯＳは、物理アドレス空間４５０に確保されたメッセージ領域４５１を、メッセージチャネルのメッセージポートの設定されたゲストＯＳ（論理パーティション）の論理パーティションアドレス空間の一方にマッピングし、他方からアンマップ（マップ解除）する処理によって、クライアントからのメッセージをサーバがアクセス可能な設定とすることでメッセージ転送を実現している。

【００７１】

図６を参照して、ゲストＯＳ間のメッセージチャネルを介したメッセージ転送シーケンスについて説明する。

【００７２】

メッセージ送信側ＯＳ（クライアント）は、ステップＳ１０１において、メッセージ受信側ＯＳ（サーバ）に対して接続要求を送出する。接続要求は、ステップＳ１０２において、制御ＯＳを介してメッセージ受信側ＯＳ（サーバ）に送信される。

【００７３】

メッセージ受信側ＯＳ（サーバ）は、接続要求の受信に基づいて、ステップＳ１０３において、接続を許可すると、メッセージ送信側ＯＳ（クライアント）とメッセージ送信側ＯＳ（クライアント）それぞれにメッセージポートの設定されたメッセージチャネルを生成する。メッセージチャネルの生成が完了すると、ステップＳ１０４において、メッセージポートに関連付けられたイベント受信ポートを介してメッセージチャネルの設定完了を各ＯＳに通知する。

【００７４】

メッセージ送信側ＯＳ（クライアント）は、ステップＳ１０５において、メッセージ領域を確保する。図４の論理アドレス空間メッセージ領域４２３である。ステップＳ１０６において、制御ＯＳは、物理アドレス空間のメッセージ領域をメッセージ送信側ＯＳ（クライアント）論理パーティションの論理パーティションアドレス空間にマップする。この処理は、図４において、ゲストＯＳ（Ａ）４２０の論理アドレス空間メッセージ領域４２３を物理アドレス空間メッセージ領域４１１にマップする処理に相当する。

【００７５】

ステップＳ１０７において、制御ＯＳは、物理アドレス空間のメッセージ領域をメッセージ送信側ＯＳ（クライアント）側論理パーティションの論理パーティションアドレス空間からアンマップ（マップ解除）するとともに、メッセージ受信側ＯＳ（サーバ）側論理パーティションの論理パーティションアドレス空間にマップする。すなわち、図４において、ゲストＯＳ（Ａ）４２０の論理アドレス空間メッセージ領域４２３の物理アドレス空間メッセージ領域４１１に対するマップを解除し、物理アドレス空間メッセージ領域４１１をゲストＯＳ（Ｂ）４３０の論理アドレス空間メッセージ領域４３３にマップする処理を実行する。

【００７６】

ステップＳ１０８において、メッセージ受信側ＯＳ（サーバ）は、メッセージ受信側Ｏ

ＯＳ（サーバ）側論理パーティションの論理パーティションアドレス空間を介して物理アドレス空間のメッセージ領域にアクセスし、メッセージを取得する。

【００７７】

ステップＳ１０９において、メッセージ受信側ＯＳ（サーバ）は、メッセージ領域を破棄する。さらに、ステップＳ１１０において、メッセージ受信側ＯＳ（サーバ）は、メッセージポートを破棄する。メッセージポートの破棄通知は、ステップＳ１１１において、イベント受信ポートを適用してメッセージ送信側ＯＳ（クライアント）側に送信され、メッセージ送信側ＯＳ（クライアント）側においてステップＳ１１２でメッセージポートを破棄することでメッセージチャネルが解消され、メッセージチャネルを介したＯＳ間通信が終了する。

【００７８】

［（２）ゲストＯＳとシステム制御ＯＳ間の通信処理］

次に、図７および図１０を参照して、システム制御ＯＳとゲストＯＳ間の通信処理について説明する。前述したように、システム制御ＯＳは、システム制御プログラムを生成し、例えばシステムに予め定められたリソース配分の上限情報などに基づいてシステム全体の動作制御を行う。

【００７９】

ゲストＯＳは、例えばゲストＯＳに設定された論理パーティションに対応するリソース要求など様々な処理のために、システム制御ＯＳに設定されたシステム制御プログラムに対してメッセージ転送を行なう。

【００８０】

前述したゲストＯＳ間の通信では、各ゲストＯＳにおいて設定される各々の論理パーティション対応の論理アドレス空間を物理アドレス空間のメッセージ領域に対してマッピング、アンマッピングする処理としてメッセージ転送を実現していた。システム制御ＯＳに設定されたシステム制御プログラムとゲストＯＳ間の通信処理においても、図７に示すように制御ＯＳ５１０の生成するメッセージチャネル５１２を適用した通信が実行されるが、システム制御ＯＳ５２０は、例えばＵＮＩＸ（登録商標）において通信に適用するソケットを生成してメッセージチャネルを介した通信を実現する。

【００８１】

図７のように、仮想ファイルを用いた通信を行う場合は、システム制御ＯＳは、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、このソケットを介してアクセス可能な仮想ファイルを作成し、物理アドレス空間のメッセージ領域を仮想ファイルを介してアクセスすることで、メッセージ送受信を実行する。

【００８２】

図１０のように、仮想ファイルを用いない通信を行う場合は、システム制御ＯＳは、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、該ソケットを介して物理アドレス空間のメッセージ領域をプロセスのアドレス空間へマッピングし、プロセスは直接メッセージ領域へのアクセスを実行する。

【００８３】

ＵＮＩＸ（登録商標）のソケットには、システム内のプロセスとの通信に使われる場合（ＵＮＩＸドメインソケット）と、ネットワークを通じた別のシステムとの通信に使われる場合（インターネットドメインソケット）がある。システム制御ＯＳは、メッセージチャネルを新たなドメインのソケットとしてシステム制御プログラムに提供する。システム制御プログラムは、既存のソケットとほぼ同じセマンティックでメッセージチャネルを用いた通信を行うことができる。

【００８４】

従来のＵＮＩＸ（登録商標）のソケット（ＵＮＩＸドメインソケット）を適用した通信処理の概要は、以下のとおりである。

ａ）メッセージ受信側（サーバ）プロセスが、ソケット（ｓｏｃｋｅｔ）システムコールでソケットを作成する。

ｂ）メッセージ受信側（サーバ）プロセスが、バインド（ｂｉｎｄ）システムコールを使用してソケットに対応する名前（ファイルディスクリプタ）を割り当てる。

ｃ）メッセージ受信側（サーバ）プロセスが、リッスン（ｌｉｓｔｅｎ）システムコールを実行し、接続準備状態に設定する。

ｄ）メッセージ送信側（クライアント）プロセスが、ソケット（ｓｏｃｋｅｔ）システムコールでソケットを作成し、コネクト（ｃｏｎｎｅｃｔ）システムコールを使用してサーバ側ソケットとの通信路を確立する。

ｅ）メッセージ受信側（サーバ）プロセスが、アクセプト（ａｃｃｅｐｔ）システムコールを実行し、メッセージ送信側（クライアント）プロセスからの要求を受け入れる。

ｆ）接続完了後に、ソケット対応のファイルディスクリプタに対する書き込み処理としてのライト（ｗｒｉｔｅ）システムコールと、読み取り処理としてのリード（ｒｅａｄ）システムコールを利用して、メッセージ送受信を実現する。

【００８５】

以上が、ＵＮＩＸにおけるソケット（ＵＮＩＸドメインソケット）を適用した通信シーケンスの概要である。本発明では、図７、図１０に示す制御ＯＳ５１０の設定する通信チャネル５１２を適用し、かつ、システム制御プログラム側では、上述した既存のソケットを適用した処理とほぼ同様の処理によってゲストＯＳとの通信を可能とする。なお、ゲストＯＳ側の処理は、前述したゲストＯＳ間通信における処理と同様の処理として実行される。

【００８６】

図７、図８（仮想ファイルを用いた通信）および図１１、図１２（仮想ファイルを用いない通信）を参照してシステム制御ＯＳ５２０と、ゲストＯＳ５３０管理の通信処理シーケンスについて説明する。なお、実際に通信を実行するのは、システム制御ＯＳ５２０の設定したシステム制御プログラムとゲストＯＳであり、通信処理には、制御ＯＳ５１０の生成する通信チャネル５１２と、システム制御プログラムカーネル５２１の生成する通信用ソケットが適用されることになる。システム制御プログラムは、様々なシステムコールを実行し、システムコールに基づく処理をカーネル５２１が実行し、カーネルにおいて通信環境の設定、通信制御処理が実行される。システム制御プログラムは、システム制御プログラムカーネル５２１の生成する通信用ソケットを適用することにより、上述したＵＮＩＸ対応のソケットを適用した通信とほぼ同様の処理で、ゲストＯＳとの通信を実行する。

【００８７】

図８および図１１は、ゲストＯＳとの通信を実行するシステム制御プログラムの処理と、カーネルの処理とを対応付けて示している。以下、各ステップ毎に処理の詳細を説明する。

［ステップＳ２０１］

まず、ステップＳ２０１において、システム制御プログラムは、接続要求を受け付けるための接続用ソケットを生成するため、ソケットシステムコール（ｓｏｃｋｅｔ（））を起動する。なお、システムコールはカーネル５２１の機能の呼出に適用される。カーネルは、ソケットシステムコール（ｓｏｃｋｅｔ（））の起動により、接続用ソケットを生成する。図７および図１０に示す接続用ソケット５２５である。

なお、生成されるソケットには、システム制御ＯＳ５２０において管理するファイルシステムに設定されるファイル名（ファイルディスクリプタ）が対応付けられる。

【００８８】

［ステップＳ２０２］

次に、システム制御プログラムは、ステップＳ２０２において、バインドシステムコール（ｂｉｎｄ（ｓｏｃｋｅｔ，・・））を起動する。カーネル５２１は、バインドシステムコール（ｂｉｎｄ（ｓｏｃｋｅｔ，・・））に基づいて、システム制御プログラム接続

要求を待つためのサービスID（ポート番号）とソケットを対応づける。各ソケットは、システム制御プログラムの提供するサービスに対応付けられて設定されることになる。

【００８９】

［ステップＳ２０３］

次に、システム制御プログラムは、ステップＳ２０３において、リッスンシステムコール（ｌｉｓｔｅｎ（ｓｏｃｋｅｔ，・・））を起動する。カーネルは、リッスンシステムコール（ｌｉｓｔｅｎ（ｓｏｃｋｅｔ，・・））に基づいて、ソケットに対応付けたサービスの接続を許可する。ソケットには、対応するサービス識別子が設定されており、システム制御プログラムは、サービス対応の通信許可設定を実行することができる。

【００９０】

［ステップＳ２０４］

次に、システム制御プログラムは、ステップＳ２０４において、アクセプトシステムコール（ｆｄ＝ａｃｃｅｐｔ（ｓｏｃｋｅｔ，・・））を起動する。カーネル５２１は、アクセプトシステムコール（ｆｄ＝ａｃｃｅｐｔ（ｓｏｃｋｅｔ，・・））に基づいて、メッセージチャネルの接続要求の有無を検証し、接続メッセージチャネルの接続要求があるかどうか調べる。

接続要求が到着していたら以下の（ａ）～（ｃ）の処理を実行する。

（ａ）通信用のソケット（図７および図１０の通信用ソケット５２６）を作成する。

（ｂ）システム制御ＯＳ５２０の管理するファイルシステムのファイル記述子（ファイルディスクリプタ：ｆｄ）テーブル（fd-table）にソケット対応のファイル記述子（ファイルディスクリプタ：ｆｄ）を登録する。

（ｃ）登録されたソケット対応のファイル記述子（ファイルディスクリプタ：ｆｄ）をシステム制御プログラムに返す。

接続要求が到着していなかったら以下の（ｄ），（ｅ）の処理を実行する。

（ｄ）プロセスをサスペンド状態にする。

（ｅ）次に接続要求が到着したときにプロセスのサスペンド状態を解除して、上記の（ａ）～（ｃ）の処理を行う。

【００９１】

以上の処理によってシステム制御プログラムとゲストＯＳ間にメッセージチャネルが設定される。図７に示す仮想ファイル５２４は、制御ＯＳ５１０の管理する物理アドレス空間メッセージ領域５１１を抽象化したファイルである。ファイルの作成処理およびファイルの削除処理は、システム制御ＯＳ５２０によって実行され、ゲストＯＳ５３０からシステム制御プログラムに対するメッセージ転送時において、メッセージ転送前は、ゲストＯＳ５３０の論理アドレス空間メッセージ領域５３３が物理アドレス空間メッセージ領域５１１にマッピングされた状態にあり、メッセージ転送後は、システム制御ＯＳ５２０側の仮想ファイル５２４として作成され、システム制御プログラムからアクセス可能な状態になる。また、システム制御プログラムからゲストＯＳ５３０に対するメッセージ転送時においては、メッセージ転送前は、システム制御ＯＳ５２０側の仮想ファイル５２４がシステム制御プログラムからアクセス可能な状態にあり、メッセージ転送後は、システム制御ＯＳ５２０が仮想ファイルを削除し、物理アドレス空間メッセージ領域５１１がゲストＯＳ５３０の論理アドレス空間メッセージ領域５３３にマッピングされる。

【００９２】

また、物理アドレス空間メッセージ領域５１１を仮想ファイル５２４を介するのではなく、直接プロセスのアドレス空間にマップする場合は、以下のように処理される。ゲストＯＳ５３０からシステム制御プログラムに対するメッセージ転送時において、メッセージ転送前は、ゲストＯＳ５３０の論理アドレス空間メッセージ領域５３３が物理アドレス空間メッセージ領域５１１にマッピングされた状態にあり、メッセージ転送後は、システム制御ＯＳ５２０が該物理アドレス空間メッセージ領域５１１をプロセスのアドレス空間にマップして、システム制御プログラムから直接アクセス可能な状態にする。また、システム制御プログラムからゲストＯＳ５３０に対するメッセージ転送時においては、メッセー

ン転送前は、システム制御ＯＳ５２０が該物理アドレス空間メッセージ領域５１１をプロセスのアドレス空間にマップして、システム制御プログラムから直接アクセス可能な状態にあり、メッセージ転送後は、システム制御ＯＳ５２０が該マッピングを解除し、物理アドレス空間メッセージ領域５１１がゲストＯＳ５３０の論理アドレス空間メッセージ領域５３３にマッピングされる。

【００９３】

なお、仮想ファイルを用いる場合、システム制御プログラムは、通信用ソケットのファイルディスクリプタを指定したリードシステムコールによって、該ソケット対応の仮想ファイル５２４の識別子を取得することができる。システム制御プログラムは、この仮想ファイルＩＤを指定したオープン（ｏｐｅｎ）、クローズ（ｃｌｏｓｅ）、リード（ｒｅａｄ）、ライト（ｗｒｉｔｅ）のいずれかのシステムコールを適用して、ソケット対応のファイルのオープン（ｏｐｅｎ）、クローズ（ｃｌｏｓｅ）、さらにメッセージの書き込み［ライト（ｗｒｉｔｅ）］、読み取り［リード（ｒｅａｄ）］が実行可能となる。

【００９４】

また、物理アドレス空間メッセージ領域５１１を仮想ファイル５２４を介するのではなく、直接プロセスのアドレス空間にマップする場合は、システム制御プログラムは、通信用ソケットのファイルディスクリプタを指定したリードシステムコールによって、物理アドレス空間メッセージ領域５１１がプロセスのアドレス空間にマップされたアドレスを取得することができる。システム制御プログラムは、このアドレスに対する直接のアクセスを行うことにより、メッセージ領域の読み取りおよび書き込みが実行可能となる。

【００９５】

システム制御プログラムは、複数のソケットを設定することが可能であり、それぞれのソケットに対応して個別の仮想ファイルが設定され、それぞれの仮想ファイルは、個別に物理アドレス空間のメッセージ領域に対応付けられ個別の通信が可能となる（仮想ファイルではなく物理アドレス空間メッセージ領域を直接プロセスのアドレス空間にマップする場合も同様）。それぞれのソケットにはサービス識別子が対応付けられており、ソケット指定のシステムコールの選択的実行により、通信許可、不許可の設定など、ソケット（サービス）毎の個別の処理が実行できる。

【００９６】

図８および図１１に示すステップＳ－Ａ～Ｅの各処理は、ステップＳ２０１～Ｓ２０４の処理によるメッセージチャネルの生成後に任意のタイミングで実行可能な処理である。各処理について説明する。

【００９７】

［ステップＳ－Ａ］

ステップＳ－Ａは、ゲストＯＳからのメッセージ受信処理である。メッセージ受信に際して、システム制御プログラムは、リードシステムコール（ｒｅａｄ（ｆｄ，・・））を起動する。カーネル５２１は、リードシステムコール（ｒｅａｄ（ｆｄ，・・））に基づいて、ゲストＯＳからのメッセージチャネルを介したメッセージ到着の有無を検証する。

仮想ファイルを用いた通信を行う場合は、メッセージが到着していたら以下の（ａ），（ｂ）の処理を実行する。

（ａ）到着したメッセージに対応する仮想ファイルを作成する。図７に示す仮想ファイル５２４である。

（ｂ）到着したメッセージ領域に対応する仮想ファイルの識別子（ＩＤ）とショートメッセージをユーザバッファにコピーする。すなわちシステム制御プログラムに通知する。システム制御プログラムは、この仮想ファイルの識別子（ＩＤ）に基づいて仮想ファイルを指定してメッセージを読み取ることができる。

メッセージが到着していなかったら、以下の（ｃ），（ｄ）の処理を実行する。

（ｃ）プロセスをサスペンド状態にする。

（ｄ）次にメッセージが到着したときにプロセスのサスペンド状態を解除して、上記（ａ），（ｂ）の処理を行う。

【００９８】
図７に示す仮想ファイル５２４は、ソケット対応のファイル記述子（ファイルディスクリプタ：ｆｄ）によって特定されるファイルであり、制御ＯＳ５１０の管理する物理アドレス空間メッセージ領域５１１に対応可能なファイルである。
【００９９】
仮想ファイルを用いない通信（物理アドレス空間メッセージ領域を直接プロセスのアドレス空間にマップする場合）では、メッセージ領域をプロセスのアドレス空間にマップし、そのアドレスとサイズをユーザバッファにコピーする。すなわちシステム制御プログラムに通知する。システム制御プログラムはこのアドレスに基づいて、メッセージ領域の内容を直接読み取ることができる。
【０１００】
［ステップＳ－Ｂ］
ステップＳ－Ｂは、メッセージに対するアクセスに関する処理である。
仮想ファイルを用いた通信を行う場合は、すなわち、仮想ファイル５２４に対する処理である。システム制御プログラムは、仮想ファイル５２４に対する処理として、オープン（ｏｐｅｎ）、クローズ（ｃｌｏｓｅ）、リード（ｒｅａｄ）、ライト（ｗｒｉｔｅ）のいずれかのシステムコールを起動することができる。カーネル５２１は、各システムコールに対応する処理を実行する。オープン（ｏｐｅｎ）、クローズ（ｃｌｏｓｅ）は、仮想ファイル５２４のオープンとクローズ処理に対応し、ライト（ｗｒｉｔｅ）は、仮想ファイル５２４に対するデータ書き込み、リード（ｒｅａｄ）は、仮想ファイル５２４からのデータ読み取りに相当する。これらの処理は、ファイル記述子を指定した処理として実行される。
【０１０１】
仮想ファイルを用いない通信（物理アドレス空間メッセージ領域を直接プロセスのアドレス空間にマップする場合）では、システム制御プログラムは、プロセスのアドレス空間にマップされたメッセージ領域を直接アクセスすることができる。
【０１０２】
［ステップＳ－Ｃ］
ステップＳ－Ｃは、メッセージの送信の際の処理である。
仮想ファイルを用いた通信の場合は、仮想ファイル５２４に書き込まれたメッセージの送信処理である。システム制御プログラムが、通信用ソケットに対して仮想ファイルのＩＤを指定したライトシステムコール（ｗｒｉｔｅ（ｆｄ，・・））を起動することで、カーネル５２１は、ライトシステムコール（ｗｒｉｔｅ（ｆｄ，・・））に基づいて、仮想ファイル５２４に書き込まれたメッセージの送信処理を実行する。
【０１０３】
制御ＯＳ５１０は、カーネル５２１が、ライトシステムコール（ｗｒｉｔｅ（ｆｄ，・・））に基づいて実行する処理に応じて、物理アドレス空間メッセージ領域５１１のシステム制御ＯＳ５２０側の仮想ファイル５２４を削除し、ゲストＯＳ５３０側の論理アドレス空間メッセージ領域５３３にマッピングする処理を実行する。この処理により、システム制御プログラムが仮想ファイル５２４を適用して書き込みを実行したメッセージが、ゲストＯＳ５３０に転送されることになる。ゲストＯＳ５３０側の処理は、前述したと同様、メッセージボート５３２、イベントボート５３１を適用した処理として実行される。
【０１０４】
仮想ファイルを用いない通信（物理アドレス空間メッセージ領域を直接プロセスのアドレス空間にマップする場合）では、システム制御プログラムが、通信用ソケットに対して物理アドレス空間メッセージ領域がプロセスのアドレス空間にマップされているアドレスを指定したライトシステムコール（ｗｒｉｔｅ（ｆｄ，・・））を起動することで、カーネル５２１は、ライトシステムコール（ｗｒｉｔｅ（ｆｄ，・・））に基づいて、物理アドレス空間メッセージ領域の送信処理を実行する。メッセージ領域のプロセスアドレス空間へのマップは、送信時に解除される。

【０１０５】
［ステップＳ－Ｄ］
ステップＳ－Ｄは、受信メッセージを削除する際の処理である。
仮想ファイルを用いた通信の場合、システム制御プログラムが、仮想ファイル５２４に対してアンリンクシステムコール（ｕｎｌｉｎｋ（））を起動することで、カーネル５２１は、アンリンクシステムコール（ｕｎｌｉｎｋ（））に基づいて、受信メッセージを特定するファイルの削除によりメッセージの破棄を実行する。
仮想ファイルを用いない通信では、単にプロセスのアドレス空間へのマップを解除し、物理アドレス空間メッセージ領域を開放する。
【０１０６】
［ステップＳ－Ｅ］
ステップＳ－Ｅは、メッセージチャネルの切断処理であり、システム制御プログラムが、ソケットを指定したクローズシステムコール（ｃｌｏｓｅ（ｓｏｃｋｅｔ））を起動することで、カーネル５２１は、メッセージポートを削除し、メッセージチャネルの切断を行なう。
【０１０７】
なお、図８、図１１には示していないが、システム制御プログラムはセレクトシステムコール（ｓｅｌｅｃｔ）によって、メッセージの送受信処理の監視が実行できる。ソケットに対応するファイルディスクリプタを特定したセレクトシステムコールを起動することで、カーネルは、セレクトシステムコール（ｓｅｌｅｃｔ）によって特定されるソケットに対する受信メッセージの有無を他の入出力処理と同様、常に監視することが可能となる。たとえば、システム制御プログラムが複数のゲストＯＳとそれぞれ異なる通信ソケットを設定した通信を行なう場合に、ゲストＯＳ（Ａ）との通信ソケット［ｆｄ＝ｓ０］と、ゲストＯＳ（Ｂ）の通信ソケット［ｆｄ＝ｓｌ］をセレクトシステムコールの対象として指定することで、カーネルは、各ソケットに対するメッセージの有無を監視し、メッセージが有る場合にシステム制御プログラムに通知する処理を行なう。このように、セレクトシステムコール（ｓｅｌｅｃｔ）の利用により、効率的な処理が可能となる。
【０１０８】
次に、図９および図１２に示すフローチャートを参照して、システム制御プログラムがゲストＯＳ側からのメッセージを受信し、その後システム制御プログラムからゲストＯＳに対してメッセージを送信する処理を実行する場合における、システム制御プログラムおよびゲストＯＳの処理について説明する。
【０１０９】
まず、ステップＳ３０１において、システム制御プログラムはソケットシステムコール（ｓｏｃｋｅｔ（））でソケットを作成する。このソケットは、図７に示す接続用ソケット５２５に相当する。
ステップＳ３０２において、システム制御プログラムはバインドシステムコール（ｂｉｎｄ（））でソケットにサービスＩＤを付与する。前述したように、各ソケットは、システム制御プログラムの提供するサービスに対応付けられて設定されることになる。
【０１１０】
ステップＳ３０３で、システム制御プログラムはリッスンシステムコール（ｌｉｓｔｅｎ（））でソケットを接続許可状態にする。システム制御プログラムは、ソケットを特定したリッスンシステムコールの発行によって、ソケット毎に接続許可状態、不許可状態の設定をすることが可能となる。
【０１１１】
ステップＳ３０４で、システム制御プログラムはアクセプトシステムコール（ａｃｃｅｐｔ（））でソケットに対する接続要求を待つ。
【０１１２】
次に、ステップＳ３０５で、ゲストＯＳがシステム制御プログラムに対してメッセージチャネルを接続する操作を行う。

この処理には、以下の処理が含まれる。
ａ）ゲストＯＳからの接続要求処理（図６のシーケンス図におけるステップＳ１０１の接続要求処理に相当）
ｂ）制御ＯＳが接続要求をシステム制御プログラムに通知し、先に説明したステップＳ３０４のアクセプトシステムコールが発行されたことに基づいて、制御ＯＳによるメッセージチャネル設定通知をゲストＯＳに通知する処理（図６に示すステップＳ１０４に相当）
以上の処理が含まれる。
【０１１３】
ステップＳ３０６で、システム制御プログラム側のカーネルによってメッセージチャネルの接続が行われると同時に、カーネルは、システム制御プログラムに対するソケットの接続処理を行う。すなわち、通信用のソケットを作成し、アクセプトシステムコール（ａｃｃｅｐｔ（））での待ち状態から復帰する。
【０１１４】
ステップＳ３０７で、ゲストＯＳはメッセージ領域を用意し、メッセージチャネルを通じて転送する。この処理には、以下の処理が含まれる。
ａ）ゲストＯＳによるメッセージ領域確保処理、すなわち、ゲストＯＳに対応する論理パーティションの論理アドレス空間にメッセージ領域を設定する処理
ｂ）ゲストＯＳの論理アドレス空間のメッセージ領域を物理アドレス空間のメッセージ領域にマッピングする処理
以上の処理が含まれる。
【０１１５】
仮想ファイルを用いた通信を行う場合は、ステップＳ３０８（ａ）で、システム制御プログラム側のカーネルは、ゲストＯＳから送られたメッセージ領域を受信し、システム制御プログラムがアクセス可能な仮想ファイルを作成する。図７に示す仮想ファイル５２４である。なお、作成される仮想ファイルには仮想ファイルＩＤが設定される。なお、制御ＯＳは、システム制御プログラム側カーネルの生成した仮想ファイル５２４に物理アドレス空間のメッセージ領域を対応付ける処理を実行する。
【０１１６】
仮想ファイルを用いない通信を行う場合は、ステップＳ３０８（ｂ）で、システム制御プログラム側のカーネルは、ゲストＯＳから送られたメッセージ領域を受信し、システム制御プログラムのアドレス空間にマップする。
【０１１７】
仮想ファイルを用いた通信を行う場合は、ステップＳ３０９（ａ）で、システム制御プログラムが通信用ソケットを指定したリードシステムコール（ｒｅａｄ（））によって、メッセージ領域に対応する仮想ファイルのＩＤを取得する。この仮想ファイルＩＤは、例えばファイル名である。
【０１１８】
ステップＳ３１０（ａ）で、システム制御プログラムは仮想ファイルを、仮想ファイルＩＤを指定したオープンシステムコール（ｏｐｅｎ（））によりオープンし、さらに、リードシステムコール（ｒｅａｄ（））により、メッセージ領域に格納されたデータを読み取る。
【０１１９】
さらに、システム制御プログラムは、オープンした仮想ファイルに対してライトシステムコール（ｗｒｉｔｅ（））を適用して、仮想ファイルにデータを書き込むことができる。
【０１２０】
仮想ファイルを用いない通信を行う場合は、システム制御プログラムは、自らのアドレス空間にマップされたメッセージ領域を直接アクセス（ステップＳ３０９（ｂ），Ｓ３１０（ｂ））すればよい。

【０１２１】
ステップＳ３１１（ａ）またはステップＳ３１１（ｂ）において、システム制御プログラムが通信用ソケットを指定して、仮想ファイルのＩＤ、もしくは物理アドレス空間メッセージ領域がマップされたプロセスアドレス空間のアドレスの書き込み処理としてのライトシステムコール（ｗｒｉｔｅ（））を実行すると、カーネルはメッセージ領域をゲストＯＳに転送する処理を行う。この処理に応じて、制御ＯＳは、システム制御プログラム側の仮想ファイルの削除、もしくはプロセスのアドレス空間にマッピングされていた物理アドレス空間のメッセージ領域をアンマップするとともに、ゲストＯＳの論理パーティションのメッセージ領域にマッピングする。この処理により、メッセージがゲストＯＳによって読み取り可能となり、ゲストＯＳによるメッセージ読み取りが実行される。
【０１２２】
上述したように、システム制御プログラムとゲストＯＳ間の通信処理においては、システム制御プログラム側は、ソケットを適用した通信プロセスを実行することで制御ＯＳの設定する通信チャネルを介したゲストＯＳとの通信が可能となる。
【０１２３】
以上、特定の実施例を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施例の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、特許請求の範囲の欄を参酌すべきである。
【０１２４】
なお、明細書中において説明した一連の処理はハードウェア、またはソフトウェア、あるいは両者の複合構成によって実行することが可能である。ソフトウェアによる処理を実行する場合は、処理シーケンスを記録したプログラムを、専用のハードウェアに組み込まれたコンピュータ内のメモリにインストールして実行させるか、あるいは、各種処理が実行可能な汎用コンピュータにプログラムをインストールして実行させることが可能である。
【０１２５】
例えば、プログラムは記録媒体としてのハードディスクやＲＯＭ（Read Only Memory）に予め記録しておくことができる。あるいは、プログラムはフレキシブルディスク、ＣＤ－ＲＯＭ(Compact Disc Read Only Memory)，ＭＯ(Magneto optical)ディスク，ＤＶＤ(Digital Versatile Disc)、磁気ディスク、半導体メモリなどのリムーバブル記録媒体に、一時的あるいは永続的に格納（記録）しておくことができる。このようなリムーバブル記録媒体は、いわゆるパッケージソフトウェアとして提供することができる。
【０１２６】
なお、プログラムは、上述したようなリムーバブル記録媒体からコンピュータにインストールする他、ダウンロードサイトから、コンピュータに無線転送したり、ＬＡＮ(Local Area Network)、インターネットといったネットワークを介して、コンピュータに有線で転送し、コンピュータでは、そのようにして転送されてくるプログラムを受信し、内蔵するハードディスク等の記録媒体にインストールすることができる。
【０１２７】
なお、明細書に記載された各種の処理は、記載に従って時系列に実行されるのみならず、処理を実行する装置の処理能力あるいは必要に応じて並列的にあるいは個別に実行されてもよい。また、本明細書においてシステムとは、複数の装置の論理的集合構成であり、各構成の装置が同一筐体内にあるものには限らない。
【産業上の利用可能性】
【０１２８】
以上、説明したように、本発明の構成によれば、複数のオペレーティングシステム（ＯＳ）が共存するマルチＯＳ環境におけるＯＳ間通信において、ＯＳ間の通信制御を実行する制御ＯＳを設定し、制御ＯＳが、通信実行ＯＳ各々に対応して設定される論理パーティ

ョン相互のメッセージ転送制御を実行する構成とし、物理アドレス空間のメッセージ領域を、メッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態から、メッセージ受信側ＯＳの論理パーティションアドレス空間のメッセージ領域へのマッピング状態に切り替える処理を実行してＯＳ間のメッセージ転送制御を行なう構成としたので、異なるＯＳ間の通信におけるスムーズな通信が可能となる。

【０１２９】

さらに、本発明の構成によれば、通信実行ＯＳが、自己の管理するファイルシステムによって識別可能なファイルディスクリプタに関連付けられたソケットを生成し、ソケットを介してアクセス可能な仮想ファイルを設定し、仮想ファイルを介してメッセージの送受信を実行する、または、物理アドレス空間のメッセージ領域をプロセスのアドレス空間にマッピングしてメッセージ送受信を実行する構成としたので、異なるＯＳとの通信において汎用的なソケットを適用したメッセージパッシングが実現される。

【０１３０】

さらに、本発明の構成によれば、ソケットに対応するサービス識別子を設定することにより、サービス対応の通信許可設定が可能となり、また、例えばシステムコールとしてのセレクトシステムコールの適用により、ソケットを介したメッセージの受信監視処理が可能となる。

【図面の簡単な説明】

【０１３１】

【図１】本発明の情報処理装置の構成例を示す図である。

【図２】プロセッサモジュールの構成例を示す図である。

【図３】本発明の情報処理装置のオペレーションシステム構成を説明する図である。

【図４】ゲストＯＳ間の通信処理について説明する図である。

【図５】メーセッジチャネルを介した通信におけるアドレスマッピングについて説明する図である。

【図６】ゲストＯＳ間の通信処理シーケンスについて説明する図である。

【図７】仮想ファイルを用いた通信の場合の、システム制御ＯＳと、ゲストＯＳ間の通信処理について説明する図である。

【図８】仮想ファイルを用いた通信の場合の、システム制御ＯＳと、ゲストＯＳ間の通信処理において実行されるシスカテムコールに基づく処理について説明する図である。

【図９】仮想ファイルを用いた通信の場合の、システム制御ＯＳと、ゲストＯＳ間の通信処理について説明するフロー図である。

【図１０】仮想ファイルを用いない通信の場合の、システム制御ＯＳと、ゲストＯＳ間の通信処理について説明する図である。

【図１１】仮想ファイルを用いない通信の場合の、システム制御ＯＳと、ゲストＯＳ間の通信処理において実行されるシスカテムコールに基づく処理について説明する図である。

【図１２】仮想ファイルを用いない通信の場合の、システム制御ＯＳと、ゲストＯＳ間の通信処理について説明するフロー図である。

【符号の説明】

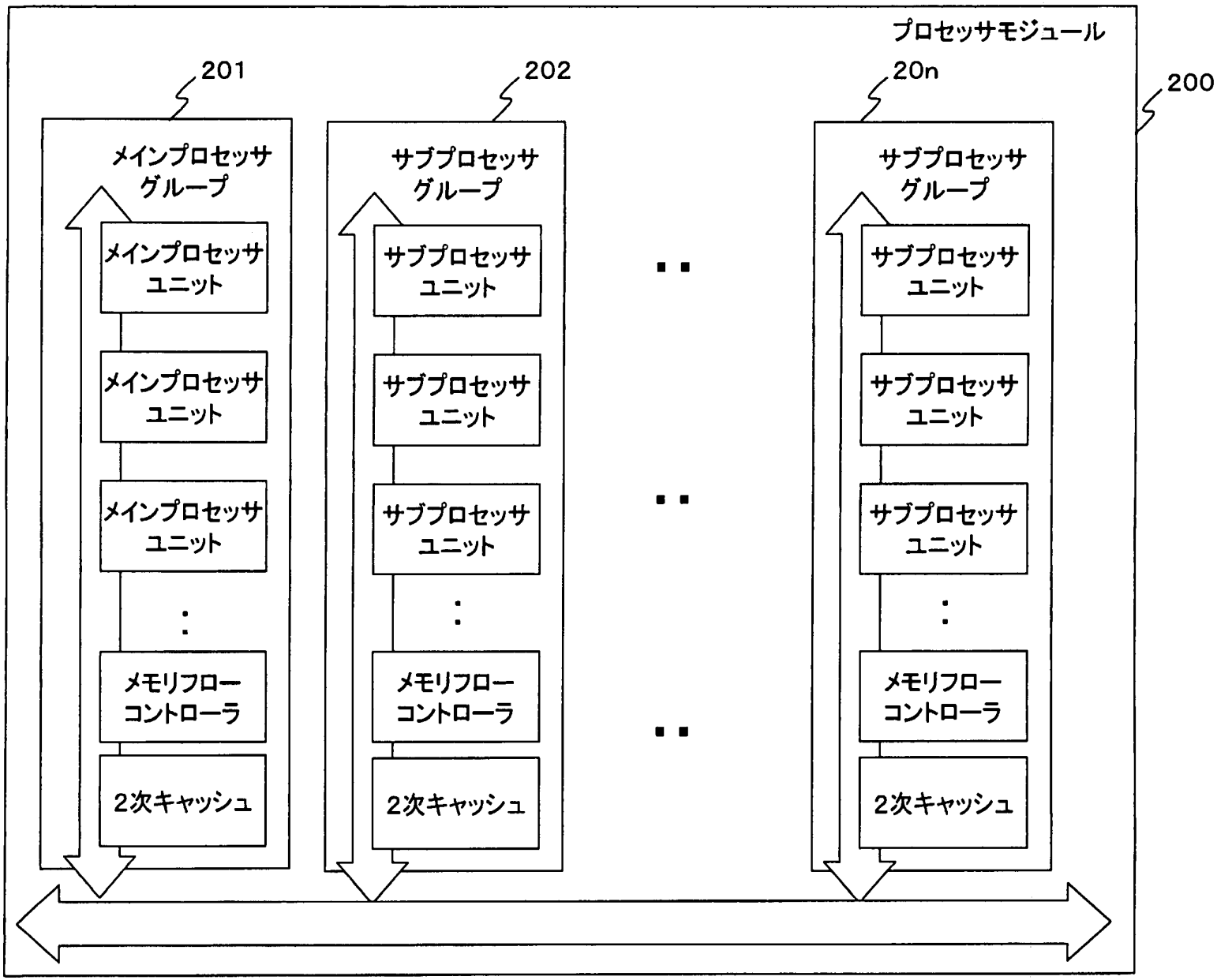
【０１３２】

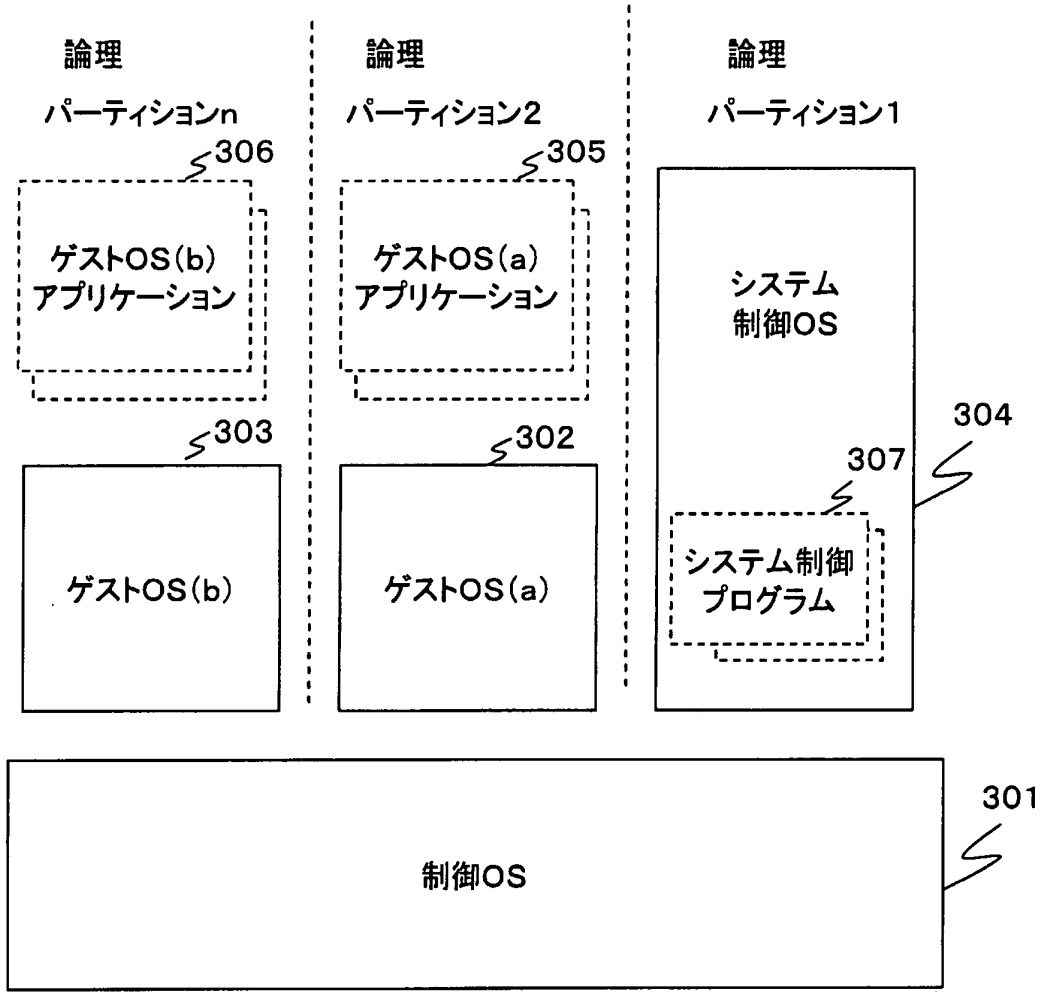
１０１　プロセッサモジュール
１０２　グラフィックエンジン
１０３　メインメモリ（ＤＲＡＭ）
１０４　ＲＯＭ
１０５　ブリッジ
１０６　コントローラ
１０７　メモリカード
１１１　ホストバス

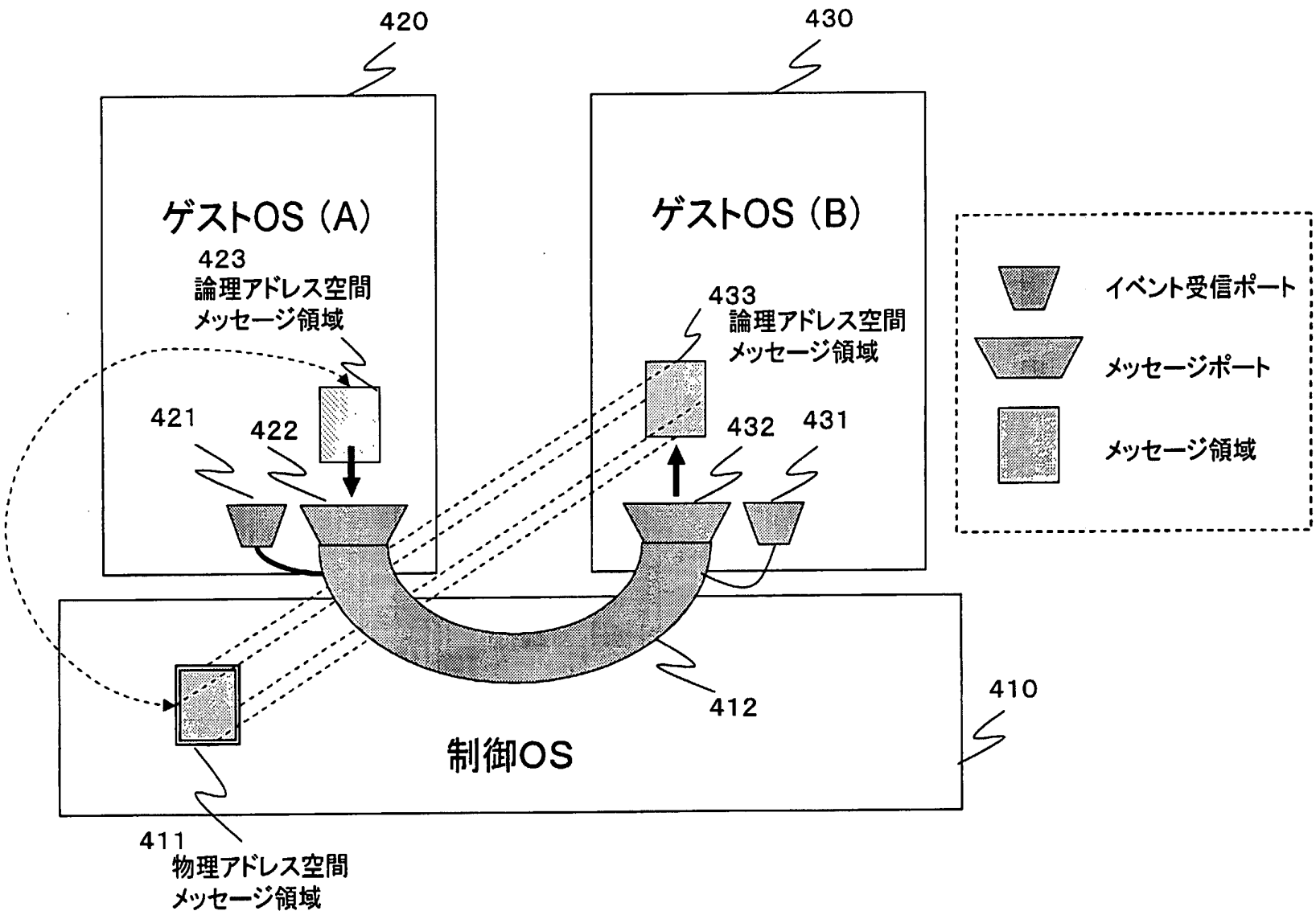
１１２　外部バス
１１３　インタフェース
１２１　入力部
１２２　出力部
１２３　ＨＤＤ
１２４　ドライブ
１２５　接続ポート
１２６　通信部
１２７　リムーバブル記録媒体
１２８　外部接続機器
２００　プロセッサモジュール
２０１　メインプロセッサグループ
２０２～２０ｎ　サブプロセッサグループ
３０１　制御ＯＳ
３０２，３０３　ゲストＯＳ
３０４　システム制御ＯＳ
３０５，３０６　ゲストＯＳアプリケーション
３０７　システム制御プログラム
４１０　制御ＯＳ
４１１　物理アドレス空間メッセージ領域
４１２　メッセージチャネル
４２０　ゲストＯＳ（Ａ）
４２１　イベント受信ポート
４２２　メッセージポート
４２３　論理アドレス空間メッセージ領域
４３０　ゲストＯＳ（Ｂ）
４３１　イベント受信ポート
４３２　メッセージポート
４３３　論理アドレス空間メッセージ領域
４５０　物理アドレス空間
４５１　メッセージ領域
４６０　ゲストＯＳ（Ａ）論理パーティションアドレス空間
４６１　メッセージ領域
４７０　ゲストＯＳ（Ｂ）論理パーティションアドレス空間
４７１　メッセージ領域
５１０　制御ＯＳ
５１１　物理アドレス空間メッセージ領域
５１２　メッセージチャネル
５２０　システム制御ＯＳ
５２１　カーネル
５２４　仮想ファイル
５２５　接続用ソケット
５２６　通信用ソケット
５２７　プロセスアドレス空間へのマップ
５３０　ゲストＯＳ（Ａ）
５３１　イベント受信ポート
５３２　メッセージポート
５３３　論理アドレス空間メッセージ領域

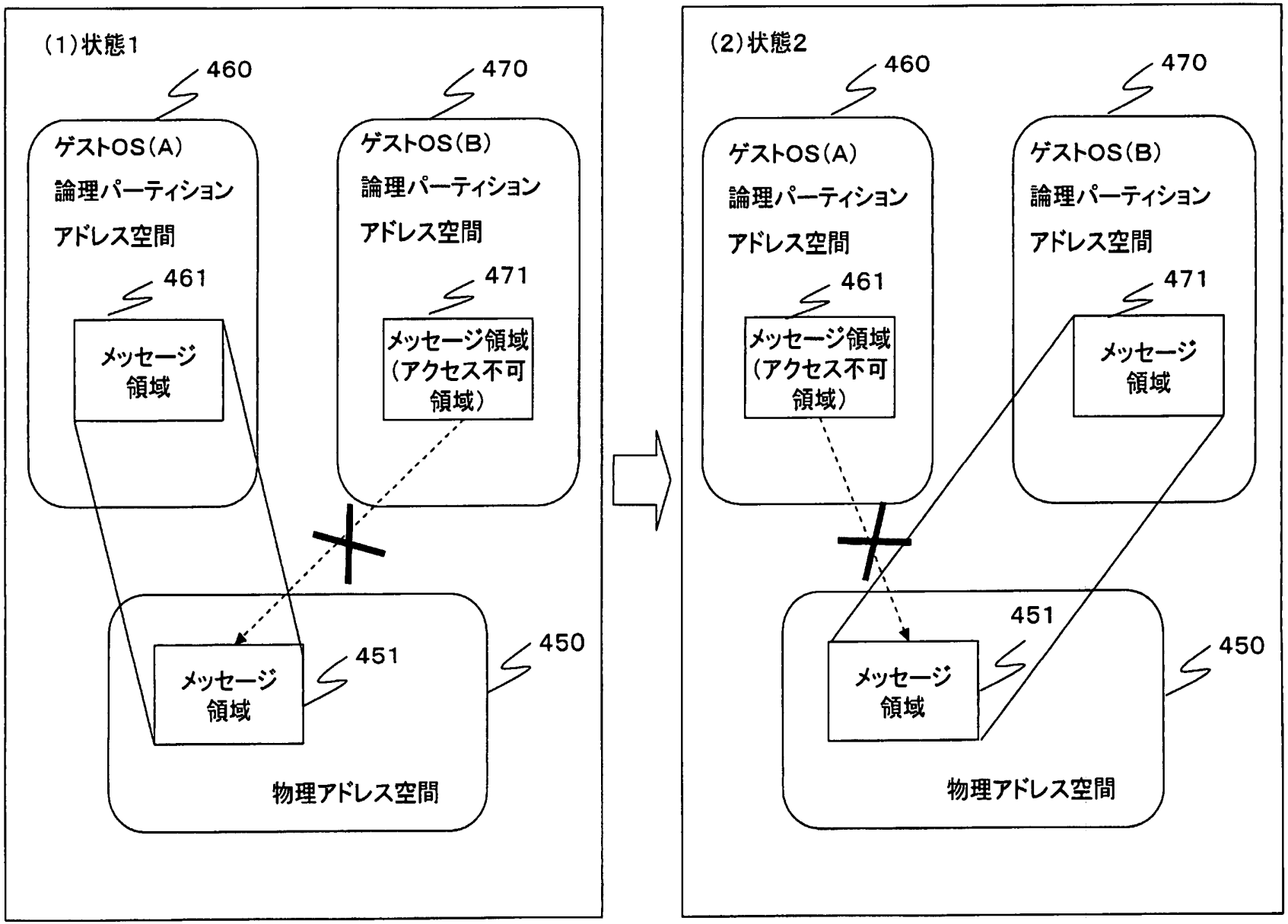
【書類名】図面
【図１】

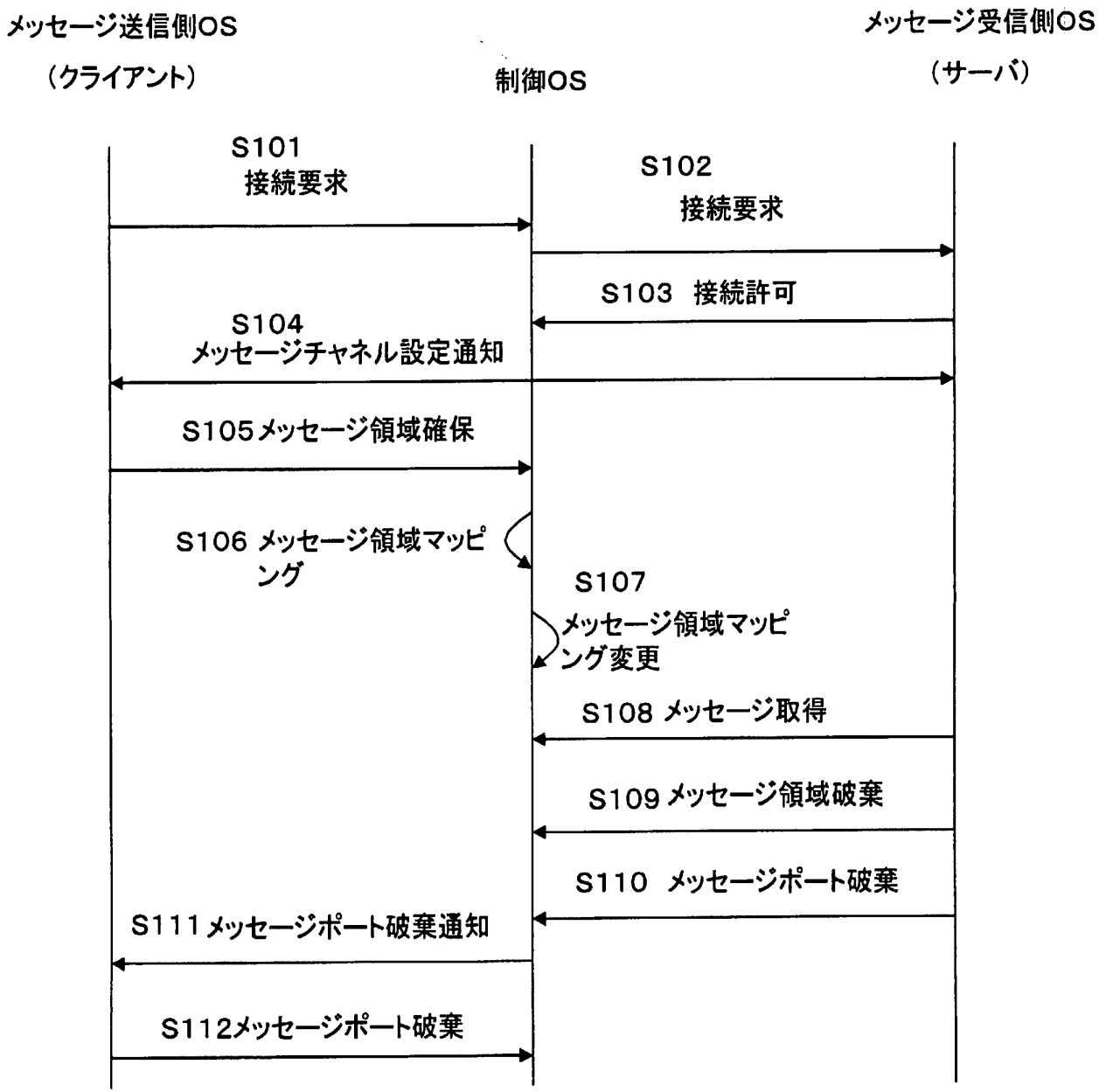
101
プロセッサ
モジュール
102
グラフィック
エンジン
103
メイン
メモリ
(DRAM)
104
ROM
105
ブリッジ
106
コントローラ
107
メモリカード
111
112
113
インタフェース
121
入力部
122
出力部
123
HDD
124
ドライブ
125
接続ポート
(USB他)
126
通信部
127
リムーバブル
記録媒体
128
外部接続機器

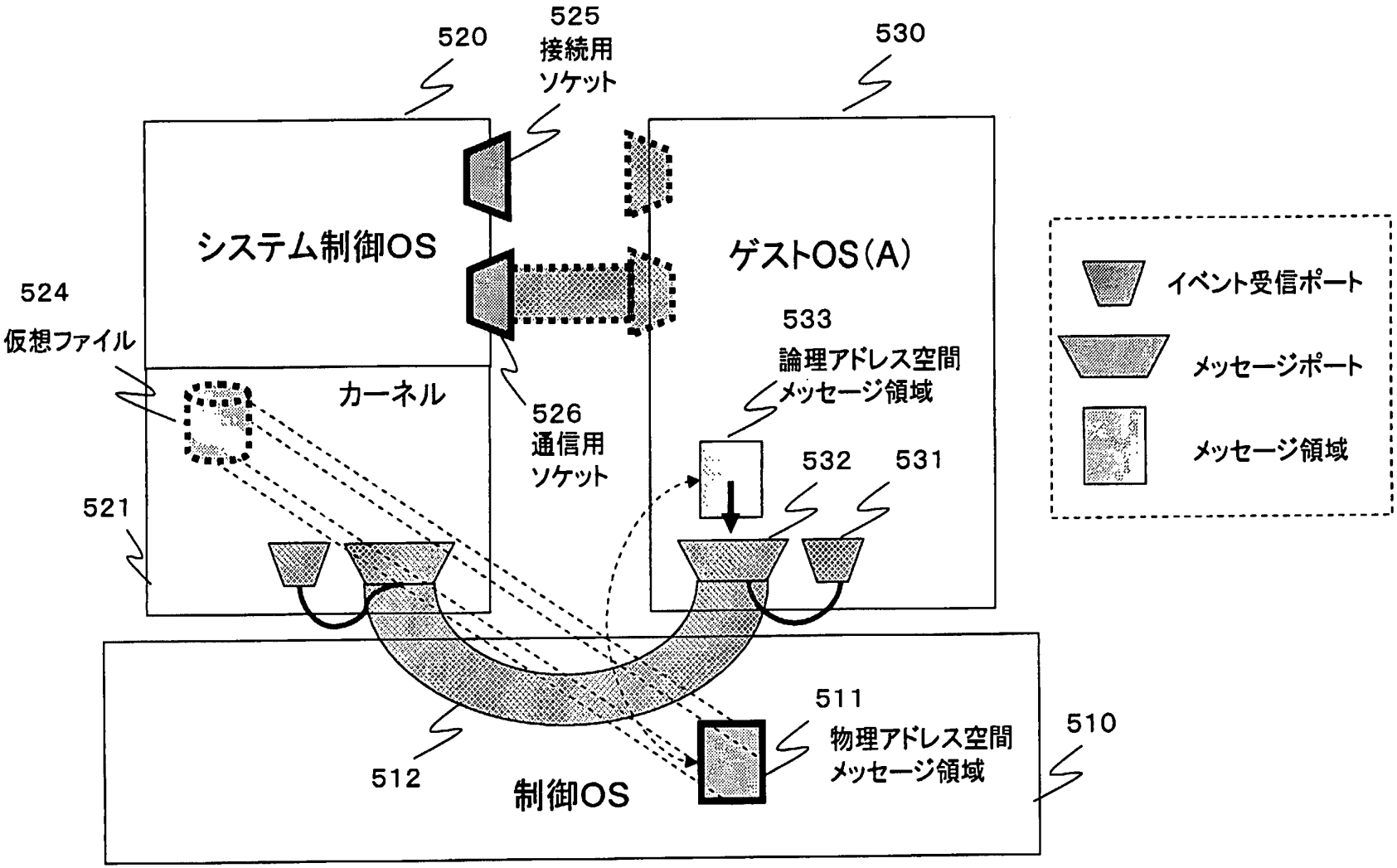

プロセッサモジュール
200
201
202
20n
メインプロセッサ
グループ
サブプロセッサ
グループ
サブプロセッサ
グループ
メインプロセッサ
ユニット
メインプロセッサ
ユニット
メインプロセッサ
ユニット
メモリフロー
コントローラ
2次キャッシュ
サブプロセッサ
ユニット
サブプロセッサ
ユニット
サブプロセッサ
ユニット
メモリフロー
コントローラ
2次キャッシュ
サブプロセッサ
ユニット
サブプロセッサ
ユニット
サブプロセッサ
ユニット
メモリフロー
コントローラ
2次キャッシュ


論理
パーティションn
306
ゲストOS(b)
アプリケーション
303
ゲストOS(b)
論理
パーティション2
305
ゲストOS(a)
アプリケーション
302
ゲストOS(a)
論理
パーティション1
システム
制御OS
304
307
システム制御
プログラム
制御OS
301

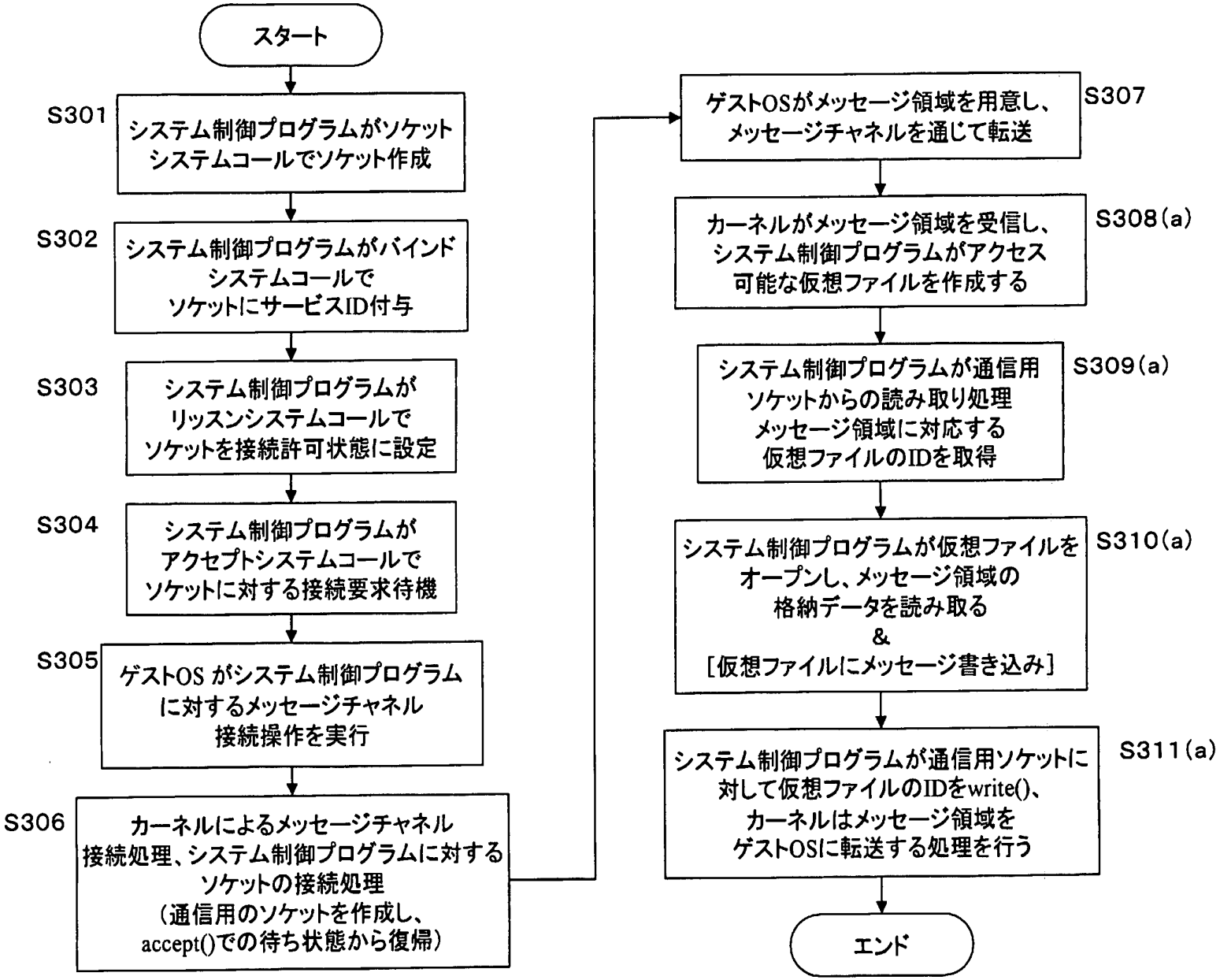

420
430
ゲストOS (A)
ゲストOS (B)
423
論理アドレス空間
メッセージ領域
433
論理アドレス空間
メッセージ領域
421
422
432
431
イベント受信ポート
メッセージポート
メッセージ領域
412
410
制御OS
411
物理アドレス空間
メッセージ領域

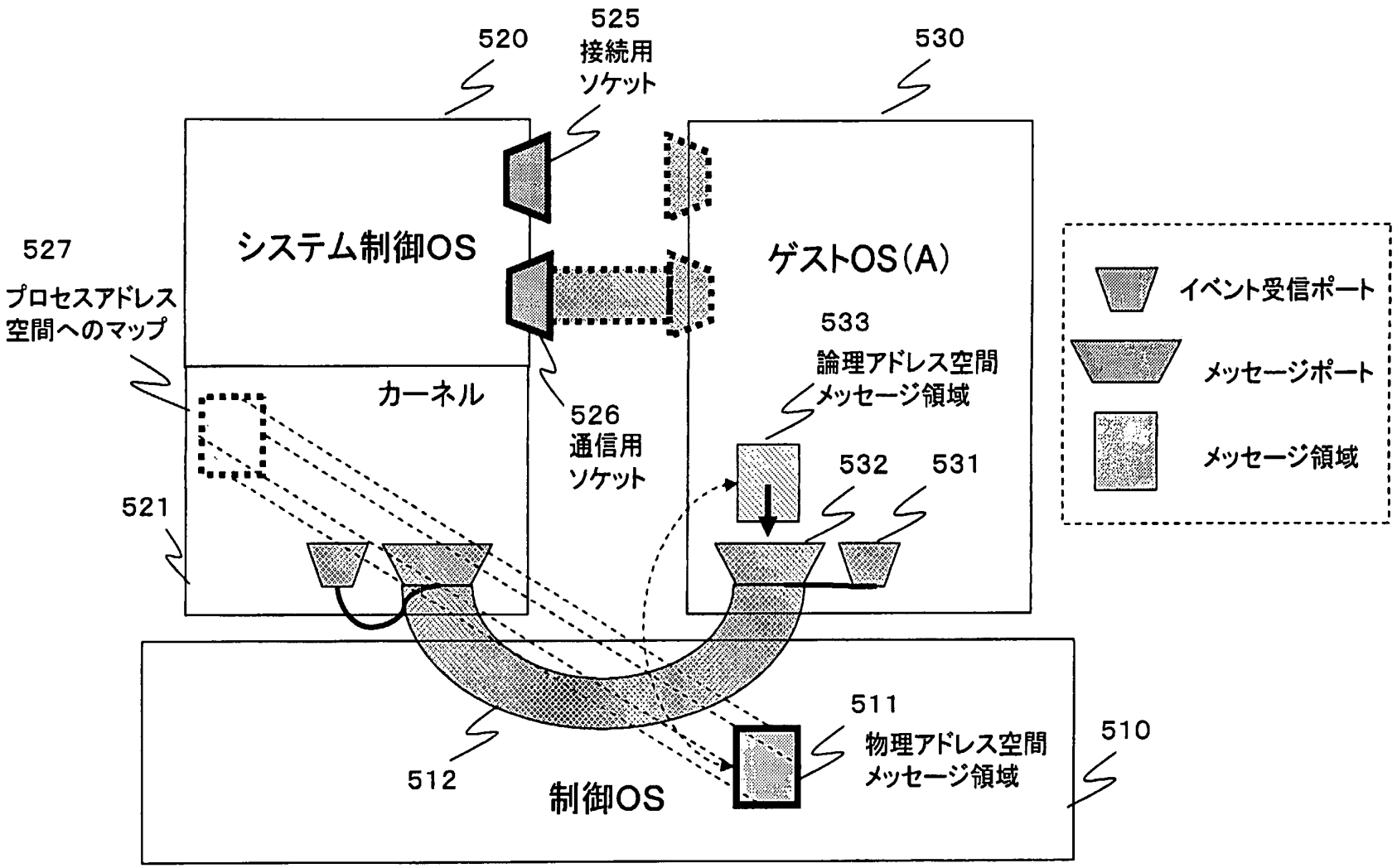

(1)状態1
460
ゲストOS(A)
論理パーティション
アドレス空間
461
メッセージ
領域
470
ゲストOS(B)
論理パーティション
アドレス空間
471
メッセージ領域
(アクセス不可
領域)
451
メッセージ
領域
450
物理アドレス空間
(2)状態2
460
ゲストOS(A)
論理パーティション
アドレス空間
461
メッセージ領域
(アクセス不可
領域)
470
ゲストOS(B)
論理パーティション
アドレス空間
471
メッセージ
領域
451
メッセージ
領域
450
物理アドレス空間


メッセージ送信側OS
（クライアント）
制御OS
メッセージ受信側OS
（サーバ）
S101
接続要求
S102
接続要求
S103 接続許可
S104
メッセージチャネル設定通知
S105メッセージ領域確保
S106 メッセージ領域マッピング
S107
メッセージ領域マッピング変更
S108 メッセージ取得
S109 メッセージ領域破棄
S110 メッセージポート破棄
S111メッセージポート破棄通知
S112メッセージポート破棄

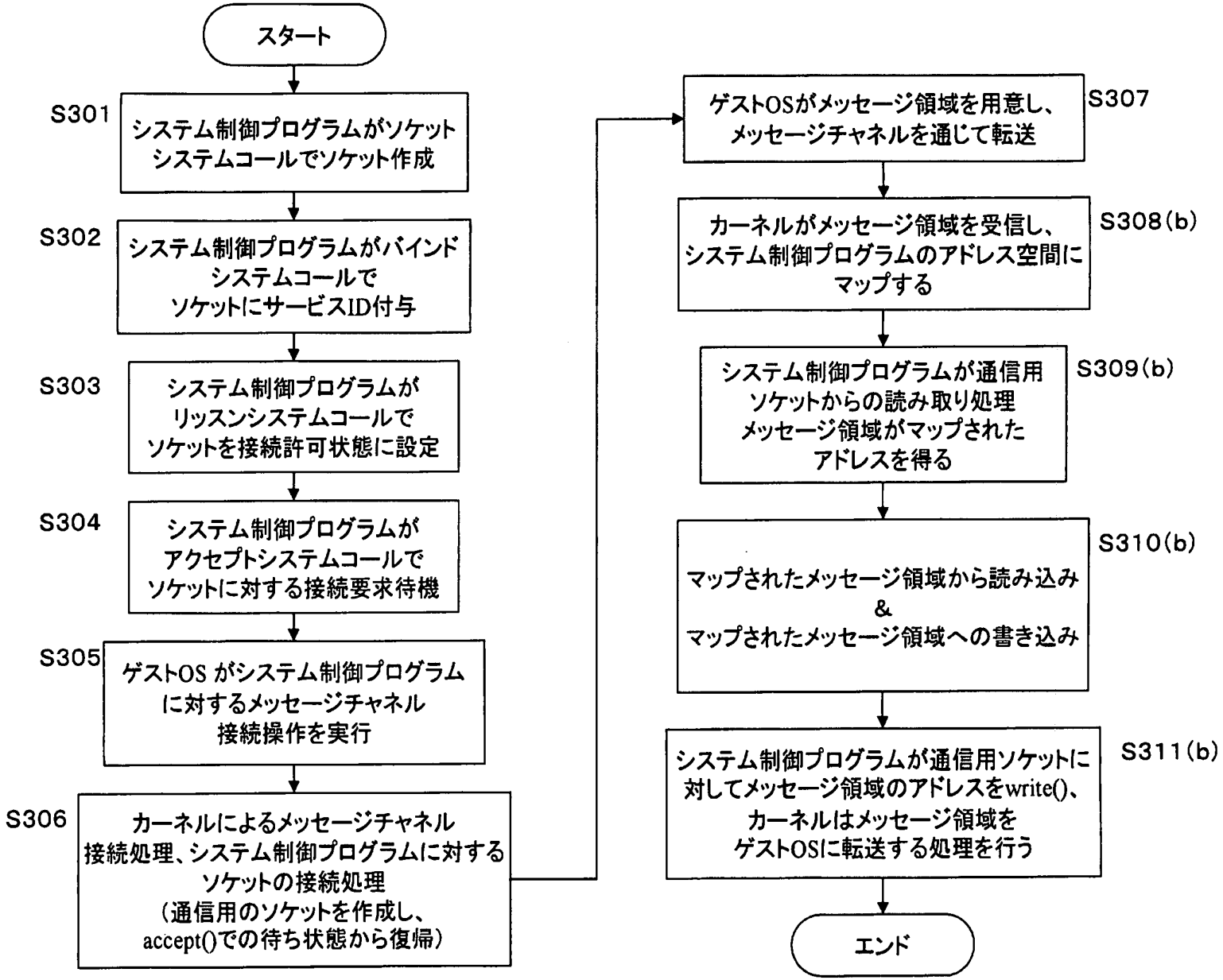

520
525
接続用
ソケット
530
システム制御OS
ゲストOS(A)
524
仮想ファイル
カーネル
526
通信用
ソケット
533
論理アドレス空間
メッセージ領域
532
531
521
511
物理アドレス空間
メッセージ領域
510
512
制御OS
イベント受信ポート
メッセージポート
メッセージ領域

	システム制御プログラムでの処理	対応するカーネル内部の処理(メッセージチャネルインタフェース)
S201	socket = socket(...)	ソケットを作成。メッセージチャネルの接続要求を受けるための準備をする。
S202	bind(socket, ..)	システム制御プログラムが接続要求を待つためのサービスIDとソケットを対応づける。
S203	listen(socket, ...)	指定したサービスへの接続を許可する。
S204	fd = accept(socket, ...)	メッセージチャネルを接続する。
S−A (a)	read(fd, ...)	メッセージを受信して対応するファイル名を得る。
S−B (a)	メッセージファイルを open/close/read/write	メッセージの内容を読み書き。
S−C (a)	write(fd, ...	指定したメッセージを転送する。
S−D (a)	メッセージファイルをunlink()	受信したメッセージを破棄する。
S−E	close(socket)	メッセージポートを削除しメッセージチャネルを切断する。


スタート
S301
システム制御プログラムがソケット
システムコールでソケット作成
S302
システム制御プログラムがバインド
システムコールで
ソケットにサービスID付与
S303
システム制御プログラムが
リッスンシステムコールで
ソケットを接続許可状態に設定
S304
システム制御プログラムが
アクセプトシステムコールで
ソケットに対する接続要求待機
S305
ゲストOS がシステム制御プログラム
に対するメッセージチャネル
接続操作を実行
S306
カーネルによるメッセージチャネル
接続処理、システム制御プログラムに対する
ソケットの接続処理
（通信用のソケットを作成し、
accept()での待ち状態から復帰）
S307
ゲストOSがメッセージ領域を用意し、
メッセージチャネルを通じて転送
S308(a)
カーネルがメッセージ領域を受信し、
システム制御プログラムがアクセス
可能な仮想ファイルを作成する
S309(a)
システム制御プログラムが通信用
ソケットからの読み取り処理
メッセージ領域に対応する
仮想ファイルのIDを取得
S310(a)
システム制御プログラムが仮想ファイルを
オープンし、メッセージ領域の
格納データを読み取る
&
[仮想ファイルにメッセージ書き込み]
S311(a)
システム制御プログラムが通信用ソケットに
対して仮想ファイルのIDをwrite()、
カーネルはメッセージ領域を
ゲストOSに転送する処理を行う
エンド


520
525
接続用
ソケット
530
システム制御OS
ゲストOS(A)
527
プロセスアドレス
空間へのマップ
533
論理アドレス空間
メッセージ領域
カーネル
526
通信用
ソケット
532
531
521
イベント受信ポート
メッセージポート
メッセージ領域
511
物理アドレス空間
メッセージ領域
510
512
制御OS

	システム制御プログラムでの処理	対応するカーネル内部の処理(メッセージチャネルインタフェース)
S201	socket = socket(...)	ソケットを作成。メッセージチャネルの接続要求を受けるための準備をする。
S202	bind(socket, ..)	システム制御プログラムが接続要求を待つためのサービスIDとソケットを対応づける。
S203	listen(socket, ...)	指定したサービスへの接続を許可する。
S204	fd = accept(socket, ...)	メッセージチャネルを接続する。
S−A (b)	read(fd, ...)	メッセージを受信して対応する(プロセスアドレス空間にマップされた)アドレスを得る。
S−B (b)	メッセージに対するアクセス	メッセージの内容を直接読み書き。
S−C (b)	write(fd, ...	指定したメッセージを転送する。
S−D (b)	メッセージの領域を開放	プロセスのアドレス空間へのマップを解除する。
S−E	close(socket)	メッセージポートを削除しメッセージチャネルを切断する。


スタート
S301
システム制御プログラムがソケット
システムコールでソケット作成
S302
システム制御プログラムがバインド
システムコールで
ソケットにサービスID付与
S303
システム制御プログラムが
リッスンシステムコールで
ソケットを接続許可状態に設定
S304
システム制御プログラムが
アクセプトシステムコールで
ソケットに対する接続要求待機
S305
ゲストOS がシステム制御プログラム
に対するメッセージチャネル
接続操作を実行
S306
カーネルによるメッセージチャネル
接続処理、システム制御プログラムに対する
ソケットの接続処理
(通信用のソケットを作成し、
accept()での待ち状態から復帰)
S307
ゲストOSがメッセージ領域を用意し、
メッセージチャネルを通じて転送
S308(b)
カーネルがメッセージ領域を受信し、
システム制御プログラムのアドレス空間に
マップする
S309(b)
システム制御プログラムが通信用
ソケットからの読み取り処理
メッセージ領域がマップされた
アドレスを得る
S310(b)
マップされたメッセージ領域から読み込み
&
マップされたメッセージ領域への書き込み
S311(b)
システム制御プログラムが通信用ソケットに
対してメッセージ領域のアドレスをwrite()、
カーネルはメッセージ領域を
ゲストOSに転送する処理を行う
エンド

【書類名】 要約書
【要約】
【課題】 オペレーティングシステム（ＯＳ）間のスムーズなメッセージ転送を実現する装置、方法を提供する。
【解決手段】 マルチＯＳ環境において、物理アドレス空間のメッセージ領域をメッセージ送信側ＯＳの論理パーティションアドレス空間のメッセージ領域から、メッセージ受信側ＯＳの論理パーティションアドレス空間のメッセージ領域へマッピング状態を切り替える処理を実行してＯＳ間のメッセージ転送制御を行なう。また、ファイルディスクリプタに関連付けられたソケットを生成し、ソケットを介してアクセス可能な仮想ファイルを設定し、仮想ファイルと物理アドレス空間のメッセージ領域のマッピングを適用したメッセージ送受信を行なう。
【選択図】 図７

０００００２１８５
19900830
新規登録
５９７０６２９９３

東京都品川区北品川６丁目７番３５号
ソニー株式会社

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/JP2005/017648

International filing date: 26 September 2005 (26.09.2005)

Document type: Certified copy of priority document

Document details: Country/Office: JP
Number: 2005-152788
Filing date: 25 May 2005 (25.05.2005)

Date of receipt at the International Bureau: 15 November 2005 (15.11.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse